Universidade Federal do Pampa
Filipe Santos Araujo

Análise de Sentimento usando a Representação Distribuída de Parágrafos para o Português

Alegrete
2015

Filipe Santos Araujo

Análise de Sentimento usando a Representação Distribuída de Parágrafos para o Português

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Fabio Natanael Kepler

Alegrete

2015

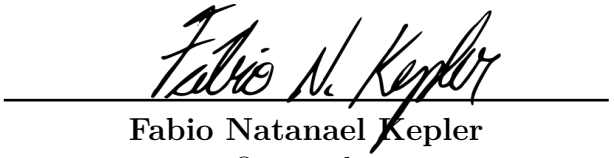
Filipe Santos Araujo

Análise de Sentimento usando a Representação Distribuída de Parágrafos para o Português

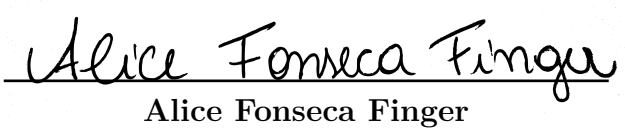
Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade Federal do Pampa como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho de Conclusão de Curso defendido e aprovado em ..... de ..... de .....

Banca examinadora:



Fabio Natanael Kepler
Orientador



Alice Fonseca Finger
UNIPAMPA

Marcelo Resende Thielo
UNIPAMPA

Resumo

Muito vem sendo discutido dentro da área de Processamento de Linguagem Natural (PLN), sobre a representação distribuída de palavras de um determinado texto. Com o contínuo crescimento de informação na internet nas últimas décadas, surge a necessidade de passar tarefas de análise desse grande volume de informação para a máquina, tarefas estas que antes eram realizadas manualmente, de modo a torná-las mais viáveis e eficientes. A representação distribuída de palavras consiste em obter uma estrutura de modelagem mais rica, que considera aspectos relevantes como ordenação, semântica e a composicionalidade das palavras de uma sentença. A dificuldade se agrava quando estas sentenças tendem a crescer no tamanho, que é o caso de textos com um grande número de parágrafos. Uma vez que se tem todas as sentenças de um determinado texto estruturadas em vetores, é possível, por exemplo, sumarizar um documento por completo, extrair sentimento, reconhecer expressões, traduzi-lo para outro idioma, dentre outros diversos tipos de tarefas. Trabalhos recentes, como o de (LE; MIKOLOV, 2014) têm apresentado técnicas como Word Vector e Paragraph Vector, que são capazes de pegar palavras, sentenças e até parágrafos e distribuí-los em vetores. Essas técnicas têm mostrado ganhos significativos em tarefas como a Análise Automática de Sentimentos (AS) e Recuperação de Informações em relação aos tradicionais modelos de linguagens utilizados como o Bag-of-Words, N-grama e Skip-grama. Esse trabalho tem como meta replicar os experimentos realizados na tarefa de AS utilizando córpus para o português brasileiro. Os experimentos realizados com o córpus em português brasileiro ReLi utilizando o método 10-fold Cross-validation atingiram uma acurácia combinada média de 82,99%. Esse resultado acima do esperado foi consequência de uma desigualdade no número de sentenças presente no córpus. Foram realizados mais experimentos com versões modificadas do ReLi buscando igualar o número de sentenças nas etapas de treinamento e teste, o que resultou numa acurácia combinada média 60,59% quando se iguala o número de sentenças com polaridade positiva e negativa.

Palavras-chave: Processamento de Linguagem Natural. Representação Distribuída de Palavras. Paragraph Vector.

Abstract

Distributed representation of words has been very discussed in the Natural Language Processing area (NLP). With the continuous growth of information on the Internet in recent decades, there is a need to in analysis tasks of this large volume of data to computers, tasks that were usually performed manually in order to make them more viable and efficient. Distributed representation of words consists of obtaining a richer modeling framework that considers relevant aspects like ordergin, semantics and compositionality of the words in a sentence. The difficulty increases when these sentences tend to grow in size, which is the case of texts with a large number paragraphs. Once you have all the sentences of a given text structured in vectors it is possible, for example, to summarize an entire document, extract sentiment, recognize expressions, translate it into another language, among other various types of tasks. Recent studies, such as (LE; MIKOLOV, 2014) have presented techniques such as Word Vector and Paragraph Vector, which are able to take words, sentences and even paragraphs and distribute them into vectors. These techniques have shown significant gains in tasks such as Automatic Sentiment Analysis (SA) and Information Retrieval over traditional language models such as Bag-of-Words, N-gram and Skip-gram. This papaer aims to replicate the experiments made in SA task using a Brazilian Portuguese corpus.The experiments performed with the ReLi corpus in Brazilian Portuguese using the 10-fold Cross-validation method achieved a average combined accuracy of 82.99%. This higher than expected result was a consequence of an unequal number of sentences in this corpus. More experiments were performed with modified versions of the ReLi in attempt to make the number of sentences equal in the training and testing stages, resulting in a average combined accuracy 60.59% when the number of sentences with positive and negative polarity are equal.

Key-words: Natural Language Processing. Distributed Representation of Words. Paragraph Vector.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Com o grande advento da internet juntamente com suas diversas mídias sociais, o interesse por parte de "entidades"em saber o que os usuários estão comentando sobre filmes, produtos, notícias e eventos em geral. Devido a essa realidade, muitos trabalhos tem sido realizados dentro da área de Análise Automática de Sentimentos (AS) na tentativa de desenvolver técnicas capazes de extrair qual a opinião de determinada pessoa por trás de seu respectivo comentário. Apesar dessa necessidade emergente, há uma escassez em trabalhos abordando a AS o idioma português brasileiro.

Uma das abordagens que vem sendo utilizadas em trabalhos recentes como solução para essa tarefa é a representação distribuída de palavras. Essa técnica consiste em pegar palavras, sentenças e até parágrafos e distribui-los em vetores, possibilitando realizar de maneira eficiente tarefas mais complexas com resultados mais precisos. Experimentos realizados por (LE; MIKOLOV, 2014) têm apresentado novos algoritmos não supervisionados, como o Paragraph Vector, que têm mostrado ganhos significativos em tarefas como a Análise Automática de Sentimentos (AS) e Recuperação de Informações em relação aos tradicionais modelos de linguagens utilizados como o Bag-of-Words, N-grama e Skip-grama. Os ganhos ao se ter textos estruturadas em um espaço vetorial são significativos, possibilitando realizar tarefas complexas como sumarização de textos, extração de informação e sentimento, tradução para outros idiomas, detecção de plágio, etc.

O trabalho de (BRUM, 2015) abordou a tarefa de AS para o português brasileiro. Para isso foram utilizadas as Redes Neurais Recursivas com Tensor (RNTN) para a tarefa de AS utilizando o córpus ReLi, disponível por /citefreitas2012vampiro. Devido a escassez de trabalhos em AS para o português, como já mencionado anteriormente, este trabalho utilizará do trabalho de Brum (2015) como base para comparar os resultados obtidos neste trabalho.tem como objetivo replicar os experimentos realizados na tarefa de AS utilizando o córpus ReLi disponível por. Sendo assim, este trabalho também utilizará o córpus ReLi na tarefa de AS utilizando o algoritmo Paragraph Vector proposto por (LE; MIKOLOV, 2014) e então comparar os resultados com os obtidos em Brum (2015).

No capítulo seguinte será realizada a fundamentação teórica, abordando modelos de linguagens que foram utilizados juntamente com os algoritmos Word Vector e Paragraph Vector. Esses modelos são o Bag-of-Words, N-grama e Skip-grama, respectivamente. Serão apresentados exemplos de seus respectivos funcionamentos, seus pontos fortes, fracos e desempenho. Ainda neste capítulo serão abordados os trabalhos relacionados e seus respectivos resultados. No terceiro capítulo será abordada a proposta do

trabalho, juntamente com a metodologia e os recursos que foram utilizados neste trabalho. Os capítulos finais serão destinados aos resultados e às discussões finais em cima dos mesmos.

2 Fundamentação Teórica

Neste capítulo será abordado o algoritmo no estado da arte Paragraph Vector, que é uma extensão do Word Vector. Ambos os algoritmos consistem na técnica de fazer a representação distribuída de palavras e sentenças de um texto em vetores, que são estruturas que proporcionam uma melhor manipulação dos dados em tarefas de análise. É importante esclarecer que ambos os algortimos não funcionam de maneira isolada, isto é, estes trabalham em conjunto com os modelos de linguagens existentes e que já são amplamente utilizados em trabalhos recentes. Sendo assim, nas seções seguintes serão abordados modelos de linguagens que são utilizados pelo Paragraph Vector.

2.1 Modelagem Bag-of-Words

Também conhecido como modelo "Saco de Palavras", trata-se de um modelo muito simples, que é um dos fatores pelo qual Bag of Words (BOW) continua sendo um dos modelos mais usados. Seu funcionamento consiste em apenas contar a ocorrência de um dado conjunto de palavras de um determinado texto ou documento, sem considerar suas ordenações ou sentidos semânticos. Ou seja, a ocorrência de palavras exatamente iguais em um texto mas que possuem sentidos diferentes, serão classificadas da mesma maneira.

Para isso, define-se um dicionário com o total de palavras presentes nos textos ou documentos analisados e atribui para cada uma delas um índice que será usado nos vetores resultantes para cada um dos textos ou documentos. As posições desse vetor, que possuem como índices os valores atribuídos às palavras presentes no dicionário, armazenarão os pesos de cada palavra com aquele índice. Os valores dos pesos podem ser calculados de dois modos: binário, que simplesmente indica se determindada palavra ocorre ou não em um texto ou documento; e baseado em frequência, que indica quantas vezes uma palavra foi contabilizada num texto ou documento, como abordado por (ALVES, 2010). Um exemplo do funcionamento do modelo BOW com os pesos calculados com base na frequência é mostrado abaixo:

Documento 1: Pedro gosta de jogar futebol. João gosta de futebol também.

Documento 2: Pedro também gosta de assistir à jogos de futebol.

Cada palavra dos textos recebe um valor de índice, como mostrado a seguir:

```
Dicionário:
{
  Pedro:    1
  gosta:    2
  de:       3
  jogar:    4
  futebol:  5
  também:   6
  assistir: 7
  à:        8
  jogos:    9
  João:    10
}
```

Esses índices então serão usados como entrada dos vetores de tamanho igual a 10 dos respectivos documentos:

Tabela 1 – Vetores de Frequência de Palavras em um Documento utilizando BOW

	1	2	3	4	5	6	7	8	9	10
Vetor do Documento 1	1	2	2	1	2	1	0	0	0	1
Vetor do Documento 2	1	1	2	0	1	1	1	1	1	0

Como pode ser visto, a modelagem BOW apresenta uma baixa complexidade, uma vez que apenas conta a ocorrência de palavras iguais em documentos, desconsiderando quaisquer outros aspectos relevantes, como ordenação e semântica das palavras. De acordo com Le e Mikolov (2014), esses aspectos são as principais fraquezas desse modelo, o que gera a necessidade de se encontrar técnicas mais bem elaboradas à medida que se deseja obter uma melhor representação de um grande volume de palavras.

2.2 Modelagem N-grama

Uma alternativa à estratégia anterior é o modelo "N-grama" (do inglês, *N-gram*), o qual elimina uma das fraquezas mencionadas na seção anterior, que é a falta de ordenação das palavras de uma dada sentença. A modelagem N-grama consiste em criar um conjunto de tokens de uma sequência de palavras, sendo que esses tokens serão agrupados de um em um, dois em dois, e assim por diante, dependendo do valor de *N*. Considerando *N=2*, chamamos o modelo de Bigrama, com *N=3*, temos o modelo Trigrama, e assim por diante. É possível se pensar no modelo N-grama como uma cadeia de Markov (GASPERIN; LIMA, 2000).

$P(X_{t+1} = s_k|X_1, ..., X_t) = P(X_{t+1} = s_k|X_t)$, sendo P a probabilidade de uma dada palavra X_{t+1} ocorrer, dado um conjunto $X_1, ..., X_t$ de palavras anteriores , apenas a primeira palavra X_t que antecede a palavra atual é levada em consideração. Para exemplificar melhor o funcionamento da modelagem N-grama, vamos considerar os exemplos a seguir:

`"O filme que passou ontem na TV foi muito bom."`

Agrupando-se a sentença utilizando o modelo N-grama com *N=1*, são obtidas as seguintes sequências:

1-grama = { O, filme, que, passou, ontem, na, TV, foi, muito, bom }

Agrupando-se a sentença utilizando o modelo n-grama com *N=2*, são obtidas as seguintes sequências:

Bigrama = { O filme, filme que, que passou, passou ontem, ontem na, na TV, TV foi, foi muito, muito bom }

Agrupando-se a sentença utilizando o modelo n-grama com *N=3*, são obtidas as seguintes sequências:

Trigrama = { O filme que, filme que passou, que passou ontem, passou ontem na, ontem na TV, na TV foi, TV foi muito, foi muito bom }

Como mostrado nos exemplos, nota-se que o problema da ordenação presente nos conjuntos da modelagem BOW foi superado. Entretanto, a questão da esparsidade dos elementos ainda é presente nessa abordagem. Além disso, percebe-se que há uma distância considerável entre o sujeito da oração "O filme" e o seu respectivo predicado "foi muito bom", impossibilitando assim fazer a ligação entre os dois tokens, mesmo adotando a modelagem trigrama.

2.2.1 Modelagem Skip-grama

Uma outra modelagem que vem sendo bastante usada na representação de sentenças é o modelo "Skip-grama" (do inglês, *Skip-gram*). Derivado da modelagem N-grama abordada na seção anterior, o Skip-grama surge na tentativa de contornar o problema da esparsidade dos dados presente nos modelos N-grama. De acordo com (GUTHRIE et al., 2006), a modelagem Skip-grama pode ser definida pelo conjunto:

$$\{w_{i_1}, w_{i_2}, ...w_{i_n} | \sum_{j=1}^{n} i_j - i_{j-1} < k\}$$

Ou seja, essa modelagem consiste em dar "saltos" de tamanho igual a k no tradicional modelo N-grama de modo a alcançar tokens mais distantes do agrupamento feito que podem ser relevantes para a análise em questão. Desse modo, o conjunto resultante terá como elementos todas as combinações de saltos possíveis até se atinja o valor de k. Para ilustrar melhor o funcionamento da modelagem Skip-grama, vamos considerar o exemplo a seguir:

```
"O filme que passou ontem na TV foi muito bom."
```

Agrupando-se a sentença utilizando o modelo N-grama, com *N=2*, são obtidas as seguintes sequências:

Bigramas = { O filme, filme que, que passou, passou ontem, ontem na, na TV, TV foi, foi muito, muito bom }

Considerando o mesmo modelo bigrama acima, mas agora aplicando um salto de tamanho igual a 1 (*k=1*) e depois outro salto de tamanho igual a 2 (*k=2*), são obtidas as seguintes sequências:

1-skip-bigrama = { O filme, filme que, que passou, passou ontem, ontem na, na TV, TV foi, foi muito, muito bom, O que, filme passou, que ontem, passou na, ontem TV, na foi, TV muito, foi bom }

2-skip-bigrama = { O filme, filme que, que passou, passou ontem, ontem na, na TV, TV foi, foi muito, muito bom, O que, filme passou, que ontem, passou na, ontem TV, na foi, TV muito, foi bom, O passou, filme ontem, que na, passou TV, ontem foi, na muito, TV bom }

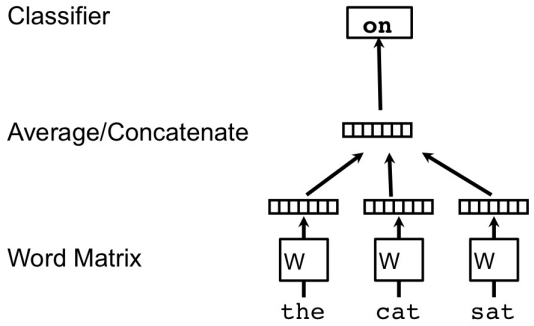
Como pode-se notar nos exemplos acima, o conjunto 1-skip-bigrama também considerou todos os elementos do conjunto bigramas, e o conjunto 2-skip-grama também considerou o conjunto 1-skip-grama, e assim seguem os demais conjuntos derivados do número do aumento de saltos.

2.3 Algoritmo Word Vector

Essa técnica nos indroduz a representação distribuída de palavras em vetores, que é o foco central desse trabalho. O Word Vector (ou "Vetor de Palavras") consiste em

estruturar cada palavra em vetores de palavras e então usá-los juntamente com modelos de linguagens apresentados nas seções anteriores. Como já foi discutido, esses modelos possuem fraquezas pelo fato de não considerarem a semântica das palavras de uma sentença, além da falta de precisão quando o tamanho das sentenças tende a crescer. Sendo assim, (SOCHER et al., 2013) usaram de redes neurais com o propósito de aprender a representação de palavras em vetores para então fazer operações de multiplicação entre essas matrizes de vetores. Em trabalhos posteriores, (MIKOLOV et al., 2013) mostraram que é menos custoso utilizar dos modelos Bag-of-Words e o Skip-grama no aprendizado de vetores de palavras, eliminando assim a complexidade gerada pelas camadas escondidas das redes neurais e também pela multiplicação de matrizes. Um exemplo pode ser visto na Figura 1.

Figura 1 – Framework para aprendizado do Word Vector



Fonte: Le e Mikolov (2014)

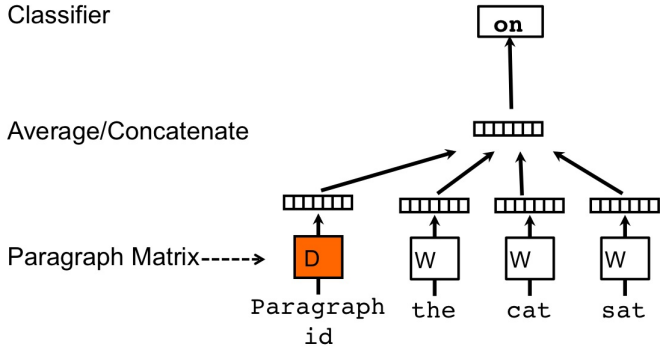
Sendo assim, o papel do Word Vector é maximizar as probabilidades das palavras de entrada e então fazer a predição da palavra em questão através da função de softmax hierárquico. Após passar por essa função, as palavras que possuem o mesmo sentido estarão mapeadas em um espaço vetorial, de modo que será possível fazer a distinção semântica entre as mesmas. Por exemplo, considerando as palavras “poderoso”, “forte” e “Paris”, será possível perceber pelos vetores de cada uma delas que “poderoso” estará mais perto de “forte” que de “Paris”. Além disso, será também permitido fazer operações algébricas entre vetores como, por exemplo, “Rei” - “homem” + “mulher” ≅ “Rainha”.

2.4 Algoritmo Paragraph Vector

Seguindo o mesmo princípio da técnica da seção anterior, o Paragraph Vector (ou “Vetor de Parágrafos”), proposto por (LE; MIKOLOV, 2014), é um algoritmo de aprendizado não supervisionado que surgiu como uma alternativa aos demais modelos devido a dificuldade dos mesmos em representar sentenças de palavras quando estas se tornam muito grandes, como textos e parágrafos. Além do mapeamento de palavras para vetor, mostrado pelo Vetor de Palavras na seção anterior, o Paragraph Vector também

faz o mapeamento dos parágrafos para vetores distintos dos de palavras, formando uma matriz de parágrafos como pode ser visto na Figura 2.

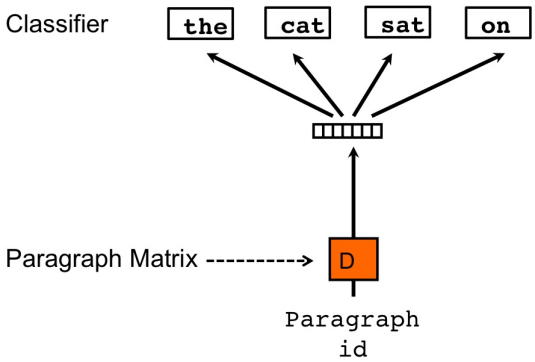
Figura 2 – Framework para aprendizado do Paragraph Vector



Fonte: Le e Mikolov (2014)

O funcionamento do Paragraph Vector é semelhante ao do Word Vector discutido anteriormente. Em resumo, esse algoritmo possui duas etapas: o treinamento para se obter a matriz com os vetores de palavras e a matriz com os vetores de parágrafos, juntamente com os pesos de softmax hierárquico; e a predição de novos parágrafos. As vantagens de se utilizar o Paragraph Vector são a consideração da ordem das palavras de uma sentença, do tamanho variável das sentenças e da semântica, características essas que não estão presentes nos modelos anteriores. Segundo (LE; MIKOLOV, 2014), o Paragraph Vector "[...] É geral e aplicável à textos de qualquer tamanho: frases, parágrafos e documentos. Ele não exige ajuste de tarefas específicas da função de pesos da palavra e nem depende de árvores sintáticas". Nesse trabalho também foi proposta uma versão alternativa do Paragraph Vector que utiliza a ideia do modelo BOW para fazer uma representação distribuída do mesmo. Sendo assim, essa versão não considera o ordenamento e o contexto das palavras de entrada para prever as palavras ao redor das mesmas na saída. Uma representação dessa versão pode ser vista na Figura 3.

Figura 3 – Versão BOW distribuído do Paragraph Vector.



Fonte: Le e Mikolov (2014)

2.5 Trabalhos Relacionados

Para testar o Paragraph Vector na tarefa de AS, (LE; MIKOLOV, 2014) realizaram dois experimentos. No primeiro, foi utilizado um dataset de reviews de filmes do site Rotten Tomatoes providenciado pela Universidade Stanford[1]. Esse córpus é constituído de 8.544 sentenças para treinamento, 2.210 para teste e 1.101 para validação. Cada sentença do dataset foi etiquetada manualmente com o sentimento presente nos reviews que variam entre muito negativo e muito positivo numa escala de 0.0 para 1.0. A tarefa então foi utilizar o Paragraph Vector em duas tarefas de classificação. A primeira delas foi considerar cinco variações de sentimento nos reviews { Very Negative, Negative, Neutral, Positive, Very Positive }, e a segunda apenas duas variações { Negative, Positive }. Como pode ser visto na Tabela 2, os resultados foram satisfatórios, superando os demais modelos propostos.

Tabela 2 – Performance do Paragraph Vector comparado a outras abordagens no dataset Treebank de Sentimento de Stanford. As taxas de erros de outros métodos estão reportadas em (SOCHER et al., 2013)

Model	Error rate (Pos/Neg)	Error rate (Fine-grained)
Naive Bayes	18.2%	59.0%
SVMs	20.6%	59.3%
Bigram Naive Bayes	16.9%	58.1%
Word Vector Averaging	19.9%	67.3%
Recursive Neural Network	17.6%	56.8%
Matrix Vector-RNN	17.1%	55.6%
Recursive Neural Tensor Network	14.6%	54.3%
Paragraph Vector	**12.2**%	**51.3**%

Fonte: Le e Mikolov (2014)

No segundo experimento foi utilizado um dataset também de review de filmes do site IMDB[2], disponibilizado por (MAAS et al., 2011). A diferença entre esse córpus e o anterior é que este possui reviews compostos por várias sentenças, o que exigiu algumas mudanças no Paragraph Vector. Esse dataset é composto por 100 mil reviews de filmes do site IMDB, que são etiquetados em apenas { Negative, Positive } e divididos em 25 mil instâncias de treinamento etiquetadas, 25 mil instâncias de teste também etiquetadas e 50 mil instâncias de testes não etiquetadas. Em vez de um classificador logístico linear, a utilização de redes neurais junto aos vetores de palavras se mostrou mais eficiente. Os resultados obtidos que evidenciam a superação do Paragraph Vector em relação aos demais modelos propostos em outros trabalhos são mostrados na Tabela 3.

[1] http://nlp.stanford.edu/sentiment/
[2] http://ai.stanford.edu/ amaas//data/sentiment/

Tabela 3 – Performance do Paragraph Vector comparado a outras abordagens no dataset do IMDB reportados por (WANG; MANNING, 2012)

Model	Error rate
BoW (bnc) (Maas et al., 2011)	12.20%
BoW (bt'c) (Maas et al., 2011)	11.77%
LDA (Maas et al., 2011)	32.58%
Full+BoW (Maas et al., 2011)	11.67%
Full+Unlabeled+BoW (Maas et al., 2011)	11.11%
WRRBM (Dahl et al., 2012)	12.58%
WRRBM+BoW (bnc) (Dahl et al., 2012)	10.77%
MNB-uni (Wang & Manning, 2012)	16.45%
MNB-bi (Wang & Manning, 2012)	13.41%
SVM-uni (Wang & Manning, 2012)	13.05%
SVM-bi (Wang & Manning, 2012)	10.84%
NBSVM-uni (Wang & Manning, 2012)	11.71%
NBSVM-bi (Wang & Manning, 2012)	8.78%
Paragraph Vector	**7.42%**

Fonte: Le e Mikolov (2014)

O trabalho de (BRUM, 2015) replicou o modelo RNTN proposto por (SOCHER et al., 2013) para a tarefa de AS para o idioma português brasileiro. O dataset utilizado foi o córpus ReLi disponível por (FREITAS et al., 2012). Como o próprio nome diz, o córpus é constituído de 12.512 sentenças resenhas de livros dividido em sentenças de polaridade positiva, negativa e neutra. Em Brum (2015), foram calculadas as acurácias simples, que englobam as sentenças de todas as polaridades e as acurácias combinadas, que apenas consideram as sentenças de polaridade positiva e negativa. O classifcador obteve uma acurácia simples média de 70,19% em sintagmas e uma acurácia combinada média de 51,36% como melhores resultados.

3 Análise de Sentimento com o Paragraph Vector

Com o recente advento das redes sociais, acompanhado de um número cada vez mais crescente de usuários ativos, surgiu o interesse de se saber, de maneira automática e precisa, quais são as opiniões por trás do que os usuários postam e comentam sobre um determinado assunto. Para que essa tarefa seja possível, faz-se necessária a utilização de modelos de linguagens que vão estruturar esses textos de modo que seja possível manipulá-los através de algoritmos de aprendizado. O trabalho de (LE; MIKOLOV, 2014) mostrou que a representação distribuída de palavras e parágrafos têm mostrado resultados no estado da arte em seus experimentos, superando trabalhos anteriores nos quais apenas foram utilizados modelos de linguagens como BOW e N-grama. Como foi discutido na seção de Trabalhos Relacionados no capítulo anterior, os métodos e métricas utilizados seguem o mesmo princípio adotado por Le e Mikolov (2014) em seu segundo experimento na tarefa de AS.

Sendo assim, para este trabalho foi escolhido como dataset o córpus ReLi (Resenha de Livros), disponível pelo site Linguateca[1]. O próximo passo foi preparar o córpus para o algoritmo Paragraph (chamado também de Doc2Vec). Devido a grande diferença entre sentenças de polaridade positiva e negativa, esse passo consistiu em, num primeiro momento, criar versões do córpus ReLi nas quais o número de sentenças com polaridade positiva sofreu alterações como será abordado no próximo capítulo. Por fim, os últimos passos foram criar casos de teste e calcular as médias das acurácias combinadas.

3.1 Recursos

Através das pesquisas realizadas ainda durante o Trabalho de Conclusão de Curso 1 foi encontrado uma implementação[2] por parte de terceiros do algoritmo do Paragraph Vector (ou Doc2Vec). Esta implementação buscou replicar os resultados obtidos por Le e Mikolov (2014) na tarefa de AS utilizando o córpus do IMDb. A proposta deste trabalho é utilizar essa implementação já disponível e adaptá-la ao córpus de língua portuguesa ReLi.

O córpus escolhido para os experimentos deste trabalhos foi o ReLi, o mesmo utilizado por Brum (2015). Com o intuito de replicar o segundo experimento realizado por citeonlineDBLP:journals/corr/LeM14 como mencionado nos trabalhos relacionados,

[1] http://www.linguateca.pt

[2] http://radimrehurek.com/gensim/models/doc2vec.html. Esse material foi criado por terceiros, e não pelos autores do artigo.

as sentenças de polaridade neutra foram desconsideradas nos experimentos. Com isso, o córpus resultou num decréscimo considerável no número de sentenças de 12.512 para 3.447. O próximo passo foi preparar o córpus, criando conjuntos de treinamento e teste, que segue o mesmo princípio adotado por Brum (2015).

3.2 Experimentos e Resultados

Nesta seção serão expostos os resultados obtidos neste trabalho juntamente com os resultados obtidos em (BRUM, 2015). Diferentemente do trabalho de Brum (2015), apenas as sentenças com polaridade positiva e negativa foram utilizadas, e as neutras foram descartadas. Para a validação do modelo foi utilizada a técnica 10-fold Cross-validation que aplica uma iteração nas sentenças do córpus no qual é selecionado 10% das sentenças iniciais do mesmo para teste e o restante para treinamento. Na próxima iteração da técnica, os próximos 10% das sentenças são selecionados para teste e o restante para treinamento, e assim sucessivamente até que se chegue ao final do córpus. Essa técnica de validação foi escolhida por apresentar um baixo custo computacional em relação aos demais de acordo com (KOHAVI et al., 1995) e (BRAGA-NETO; DOUGHERTY, 2004). Para facilitar as etapas de treinamento e teste que serão abordadas a seguir, as versões do córpus foram dividas em dois arquivos principais, sendo um destinado a armazenar as sentenças de polaridade positiva e outro para as de polaridade negativa.

A cada execução do Doc2Vec são gerados arquivos de log contendo as polaridades inferidas pelo algoritmo, as polaridades previamente anotadas para fazer comparações e as respectivas matrizes de confusão. As matrizes de confusão seguem o formato descrito na Tabela 4.

Tabela 4 – Matriz de Confusão

		Anotado	
		Positivo	**Negativo**
Inferido	**Positivo**	VP	FP
	Negativo	FN	VN

VP = Verdadeiros Positivos, FP = Falsos Positivos, FN = Falsos Negativos e VN = Verdadeiros Negativos

Sendo assim, os casos de treinamento e teste foram executados como descrito previamente e com os dados gerados nas matrizes foram calculadas as médias das acurácias entre os dez casos de testes gerados. Os valores das acurácias médias foram calculados como mostrado a seguir:

$$Acurácia = A = \frac{VP + VN}{P + N},$$

onde P corresponde ao total de sentenças positivas e N negativas

$$Média = \frac{\sum_{i=1}^{10} A_i}{10}$$

Quando os casos de treinamento e teste foram executados apenas sobre as sentenças de polaridade positiva e negativa do córpus ReLi, notou-se uma acurácia combinada média de 82,99%, o que a princípio pareceu ser muito alta. Com o intuito de obter uma melhor compreensão do resultado obtido, foram geradas as matrizes de confusão de cada caso gerado pelo método 10-fold Cross-validation. Com isso, foi observado que devido a essa grande diferença entre a quantidade de sentenças, o algoritmo Doc2Vec obtém essa alta acurácia simplesmente adivinhando grande parte das polaridades das sentenças de teste como positivas, o que faz com que a taxa de erro diminuia uma vez que há muito mais sentenças com polaridade positiva. A média dos valores obtidos pelas matrizes de confusão geradas pelos casos de treinamento e teste para cada versão do córpus ReLi são mostradas em Tabela 5, Tabela 6, Tabela 7 e Tabela 8.

Tabela 5 – Média dos valores das Matrizes de Confusão para o ReLi v1

		Inferido	
		Positivo	**Negativo**
Anotado	**Positivo**	270,5	14,5
	Negativo	53,5	5,5

Tabela 6 – Média dos valores das Matrizes de Confusão para o ReLi v2

		Inferido	
		Positivo	**Negativo**
Anotado	**Positivo**	56	3
	Negativo	53,5	5,5

Tabela 7 – Média dos valores das Matrizes de Confusão para o ReLi v3

		Inferido	
		Positivo	**Negativo**
Anotado	**Positivo**	33,5	25,5
	Negativo	27	32

Tabela 8 – Média dos valores das Matrizes de Confusão para o ReLi v4

		Inferido	
		Positivo	**Negativo**
Anotado	**Positivo**	52,5	6,5
	Negativo	47,5	11,5

Como pode ser observado nos dados presentes na matriz de confusão acima, o modelo do Paragraph Vector infere 324 (270+54) polaridades de sentenças como positivas, o que representa aproximadamente 94% do total das sentenças destinadas a etapa de teste. Sendo assim, devido ao fato do número de sentenças com polaridade positiva ser muito maior que o de negativa (aproximadamente cinco vezes maior), algumas modificações foram feitas no mesmo com o propósito de se obter uma comparação mais precisa entre os resultados obtidos. Para isso, foram criadas três versões do córpus ReLi: 1) A primeira versão (ReLi v1) manteve o córpus com a quantidade de sentenças com polaridade positiva e negativa inalterada tanto para treinamento e teste; 2) na segunda versão (ReLi v2) foram reduzidas as sentenças com polaridade positiva para teste de modo a igualar à quantidade de sentenças com polaridade negativa para teste; 3) na terceira versão (ReLi v3) foram reduzidas as sentenças com polaridade positiva de modo a igualar a quantidade de sentenças de ambas as polaridades para treinamento e teste; e 4) uma última versão (ReLi v4) com o dobro de sentenças com polaridade positiva para treinamento em relação as negativas. A Tabela 9 ilustra as versões do córpus criadas com suas respectivas quantidades de sentenças.

Tabela 9 – Versões do córpus ReLi

	Treinamento		**Teste**	
	Positivo	Negativo	Positivo	Negativo
ReLi v1	2569	534	285	59
ReLi v2	2795	534	59	59
ReLi v3	534	534	59	59
ReLi v4	1068	534	59	59

Depois de criadas as quatro versões do córpus ReLi, a técnica 10-fold Cross-validation foi repetida para cada versão e os arquivos de log foram gerados como comentado anteriormente. Os valores das acurácias médias são mostrados na Tabela 10.

Tabela 10 – Média das acurácias

Versão do córpus	**Acurácia**
ReLi utilizando RNTN (BRUM, 2015)	51.36%
ReLi v1 utilizando Doc2Vec	82,99%
ReLi v2 utilizando Doc2Vec	53,81%
ReLi v3 utilizando Doc2Vec	60,59%
ReLi v4 utilizando Doc2Vec	59,74%

Como pode ser observado, as acurácias das versões modificadas sofreram quedas significativas, o que era esperado. Ainda assim, as acurácias se mantiveram acima da acurácia combinada atingida por (BRUM, 2015). Vale ressaltar que, apesar de em ambos os trabalhos terem levado em conta a acurácia combinada para comparação entre os resultados, houve uma diferença na etapa de treinamento e teste. Em (BRUM, 2015),

as matrizes de confusão possuíam uma tupla extra correspondente as sentenças neutras. Apesar do cálculo da acurácia combinada apenas considerar as tuplas correspondentes às polaridades positivas e negativas, não podemos descartar que o modelo inferiu sentenças anotadas com polaridade neutra como positiva e negativa, e vice-versa. Por esse fator, não foi possível fazer uma comparação mais precisa, uma vez que o modelo Doc2Vec usado apenas aceita polaridades binárias.

4 Discussões Finais

Como foi abordado nos capítulos anteriores, a representação distribuída de palavras consiste em obter uma estrutura de modelagem mais completa, que considera aspectos relevantes como ordenação, semântica e a composicionalidade das palavras de uma sentença. A dificuldade se agrava quando estas sentenças tendem a crescer no tamanho, que é o caso dos parágrafos. Sendo assim, o algoritmo Paragraph Vector tem como objetivo considerar não apenas o mapeamento das palavras em si, mas também dos seus respectivos parágrafos. Uma vez que se tem todos os parágrafos de um dado texto em vetores, análises semânticas mais complexas em cima dessas palavras possuem resultados satisfatórios.

Sendo assim, o foco desse trabalho foi explorar o Paragraph Vector juntamente com os modelos de linguagens abordados anteriormente na tarefa de AS para o português brasileiro. O trabalho de (LE; MIKOLOV, 2014) apresentou resultados no estado da arte na tarefa de AS utilizando o algoritmo Paragraph Vector. Uma outra proposta utilizando RNTNs também na tarefa de AS foi apresentada por (BRUM, 2015), que obteve resultados promissores que serviram de base para comparação com os deste trabalho.

Tendo em vista o fato da proporção de sentenças com polaridade positiva ser muito maior que as de polaridade negativa, os resultados obtidos com as versões criadas do córpus ReLi foram satisfatórios quando comparados com o trabalho de (BRUM, 2015) no qual foi utilizado o mesmo dataset. Infelizmente experimentos mais precisos envolvendo as sentenças de polaridade neutra não foram possíveis devido à limitações no modelo utilizado para o experimento como comentado no capítulo anterior certamente causariam impacto sobre os valores das acurácias médias. Já em relação ao trabalho de (LE; MIKOLOV, 2014) os resultados se mostraram inferiores, uma vez que atingiram uma acurá 92,58%. Isso se deve pelo fato do tamanho do córpus IMDB ser aproximadamente oito vezes maior que o ReLi.

Como trabalhos futuros sugere-se que seja utilizado um córpus com uma quantidade de sentenças não só maior do que neste trabalho como também balanceado de modo que as matrizes de confusão não obtenham valores desiguais. Outra sugestão é incluir mais polaridades além das positivas e negativas para permitir comparações mais precisas com trabalhos como o de Brum (2015).

Referências

ALVES, A. I. M. Modelo de representação de texto mais adequado à classificação. Instituto Politécnico do Porto. Instituto Superior de Engenharia do Porto, 2010. Disponível em: <http://recipp.ipp.pt/bitstream/10400.22/1908/1/DM_AlexandraAlves_2010_MEI.pdf>. Citado na página 15.

BRAGA-NETO, U. M.; DOUGHERTY, E. R. Is cross-validation valid for small-sample microarray classification? *Bioinformatics*, Oxford Univ Press, v. 20, n. 3, p. 374–380, 2004. Citado na página 24.

BRUM, H. B. Análise de sentimentos para português brasileiro usando redes neurais recursivas. p. 5, 2015. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/tilic/2015/002.pdf>. Citado 6 vezes nas páginas 13, 22, 23, 24, 26 e 29.

FREITAS, C. et al. Vampiro que brilha... rá! desafios na anotaçao de opiniao em um corpus de resenhas de livros. *ENCONTRO DE LINGUÍSTICA DE CORPUS*, v. 11, 2012. Citado na página 22.

GASPERIN, C. V.; LIMA, V. L. S. Fundamentos do processamento estatístico da linguagem natural. *Trabalho Individual, PUC-RS*, 2000. Disponível em: <http://www3.pucrs.br/pucrs/files/uni/poa/facin/pos/relatoriostec/tr021.pdf>. Citado na página 16.

GUTHRIE, D. et al. A closer look at skip-gram modelling. In: *Proceedings of the 5th international Conference on Language Resources and Evaluation (LREC-2006)*. [s.n.], 2006. p. 1–4. Disponível em: <http://homepages.inf.ed.ac.uk/ballison/pdf/lrec_skipgrams.pdf>. Citado na página 17.

KOHAVI, R. et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Ijcai*. [S.l.: s.n.], 1995. v. 14, n. 2, p. 1137–1145. Citado na página 24.

LE, Q. V.; MIKOLOV, T. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. Disponível em: <http://arxiv.org/abs/1405.4053>. Citado 10 vezes nas páginas 4, 5, 13, 16, 19, 20, 21, 22, 23 e 29.

MAAS, A. L. et al. Learning word vectors for sentiment analysis. In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011. (HLT '11), p. 142–150. ISBN 978-1-932432-87-9. Disponível em: <http://dl.acm.org/citation.cfm?id=2002472.2002491>. Citado na página 21.

MIKOLOV, T. et al. Distributed representations of words and phrases and their compositionality. In: BURGES, C. et al. (Ed.). *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013. p. 3111–3119. Disponível em: <http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf>. Citado na página 19.

RINO, L. H. M.; PARDO, T. A. S. A sumarização automática de textos: principais características e metodologias. In: *Anais do XXIII Congresso da Sociedade Brasileira de Computação*. [s.n.], 2003. v. 8, p. 203–245. Disponível em: <http://www.icmc.usp.br/~taspardo/JAIA2003-RinoPardo.pdf>. Nenhuma citação no texto.

SOCHER, R. et al. Semantic compositionality through recursive matrix-vector spaces. In: *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. (EMNLP-CoNLL '12), p. 1201–1211. Disponível em: <http://dl.acm.org/citation.cfm?id=2390948.2391084>. Nenhuma citação no texto.

SOCHER, R. et al. Recursive deep models for semantic compositionality over a sentiment treebank. In: CITESEER. *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*. 2013. v. 1631, p. 1642. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.383.1327&rep=rep1&type=pdf>. Citado 4 vezes nas páginas 9, 19, 21 e 22.

VILLAVICENCIO, A. et al. Identificação de expressões multipalavra em domínios específicos. *Linguamática*, v. 2, n. 1, p. 15–33, 2010. Disponível em: <http://www.linguamatica.com/index.php/linguamatica/article/viewFile/43/57>. Nenhuma citação no texto.

WANG, S.; MANNING, C. D. Baselines and bigrams: Simple, good sentiment and topic classification. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012. (ACL '12), p. 90–94. Disponível em: <http://dl.acm.org/citation.cfm?id=2390665.2390688>. Citado 2 vezes nas páginas 9 e 22.