UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE ESTATÍSTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM ESTATÍSTICA

LUÍS GUSTAVO SILVA E SILVA

COWORDS: A PROBABILISTIC MODEL FOR TEXT VISUALIZATION

Belo Horizonte
2017

LUÍS GUSTAVO SILVA E SILVA

COWORDS: A PROBABILISTIC MODEL FOR TEXT VISUALIZATION

Tese apresentada ao Programa de Pós-Graduação em Estatística do Departamento de Estatística da Universidade Federal de Minas Gerais como parte dos requisitos para a obtenção do grau de Doutor em Estatística.

Orientador: Prof. Dr. Renato M. Assunção

Belo Horizonte
2017

Para meus pais Wilson e Dorcina pelo incentivo e amor incondicional.
Para Priscila pela compreensão e amor.

Agradecimentos

Ao Professor Renato Assunção, orientador e amigo. Sou grato por todos os ensinamentos que foram transmitidos em nossas reuniões semanais e pelo incentivo durante todo o período do doutorado.

Ao Professor Osvaldo Carvalho, pela oportunidade de trabalhar em diversos projetos desde o mestrado os quais contribuíram para minha formação acadêmica e profissional.

Ao Professor Martin Ester, por todo aprendizado adquirido durante o período em que fiquei na Simon Fraser University.

Aos professores da Pós-Graduação em Estatística da UFMG por contribuírem na minha formação.

Aos funcionários do Departamento por estarem sempre dispostos a nos atender, em especial à Rogéria.

Aos colegas da Pós-Graduação, pela amizade e companheirismo ao longo destes anos, em especial ao amigo Rodrigo, por sua disposição em ajudar em qualquer situação.

Aos amigos do Laboratório de Estatística Espacial - LESTE, vocês foram essenciais nesta caminhada, ora como estatísticos, ora como psicólogos. Muito obrigado Larissa, Douglas, Bruno, Milton e a nossa ex-integrante, Raquel pelo companheirismo.

Aos amigos do grupos Stats4Good, por compartilharem seus conhecimentos. Em especial, Augusto e Godoy pela dedicação.

Aos amigos de infância e de graduação que deram sentido a esta caminhada: Bruno, Bruno Vidigal, Iago, Laura, Lucas, Marcelo, Marconi e Samuel. Em especial ao Roberto, pelas discussões acadêmicas e também pelas discussões filosóficas sobre a vida. Você com certeza é uma grande inspiração.

Aos meus pais e irmãos por sempre me incentivarem, mesmo não sabendo o que faço (risos). Obrigado por me ensinarem a tratar as adversidades da vida sempre com o bom humor característico da Família Silva e Silva. Obrigado pelo carinho e amor incondicional de vocês. Aos meus afilhados, Bernardo e Pedrinho pelas brincadeiras.

À Priscila, agradeço por todo seu amor, carinho e cuidado. Sem eles, com certeza esta jornada não poderia ser concluída. Obrigado por acreditar em todos os meus sonhos e por ter ajudado a realizà-los. Muito outros sonhos ainda serão vividos.

À Deus pela minha saúde e paz de espírito concedido ao longo de toda minha vida.

Ao apoio financeiro da CAPES.

Resumo

Nesta tese, é introduzido o algoritmo COWORDS, um novo algoritmo estocástico para criação de múltiplas nuvens de palavras, uma nuvem para cada documento. As palavras, que são compartilhadas em múltipos documentos e possuem relevância nestes documentos, são colocadas na mesma posição em todas as nuvens. Portanto, documentos de textos similares produzem nuvens similares e compactas, facilitando a comparação. COWORDS é baseado em uma distribuição de probabilidade em que as configurações mais prováveis de serem obervadas desta distribuição são aquelas que seguem os princípios: *tightness*: as palavras que formam a nuvem devem ficar o mais próximas uma das outras; *overlapping*: as palavras não podem se sobrepor em todas as nuvens; *position*: as palavras que são compartilhadas pelas múltiplas nuvens deverão aparecer sempre na mesma posição. Configurações que não seguem estes princípios tem uma probabilidade baixa de serem observadas. Para selecionar amostras de configurações desta distribuição utilizamos métodos de *Markov Chain Monte Carlo* (MCMC). Uma extensão do COWORDS para geração de múltiplas nuvens de palavras que leva em consideração a semântica das palavras também é introduzida nesta tese. Portanto, palavras que são semânticamente correlacionadas deverão ficar próximas uma das outras em todas as nuvens, com isso adicionamos mais um princípio chamado *semantic*. Vários estudos de simulação, bem como estudos de casos são realizados para avaliar e demonstrar a eficácia do algoritimo COWORDS.

Palavras-chave: nuvem de palavras, visualização de textos, visualização semântica, busca estocástica.

Abstract

This thesis introduces COWORDS, a new stochastic algorithm to create multiple word clouds, one for each document. The shared words in multiple documents are placed in the same position in all clouds. Similar documents produce similar and compact clouds, making easier to compare and interpret simultaneously several word clouds. The algorithm is based on a probability distribution in which the most probable configurations are those with a desirable visual aspect, such as a low value for the total distance between the words in all clouds. The visual aspect and the probabilistic model are guided by three principles: (i) *tightness*: it requires that the returned configurations should have all clouds with a minimum empty space amount between the words; (ii) *overlapping*: the words in each cloud must have no overlap; (iii) *position*: the words must be in the same spatial location in each cloud where they appear. The word configurations that do not follow these principles have a low probability of being observed. We built a Metropolis-Hastings algorithm, a special case of a Markov Chain Monte Carlo (MCMC) simulation method, to sample from the proposed clouds probability distribution. Our algorithm can easily incorporate additional constraints besides requiring the same position of the words in the different clouds. In addition, an extension of COWORDS is proposed. This extension allows the COWORDS algorithm to generate temporal word clouds preserving the semantic position of the words across all clouds. This new feature keeps the three main principles of COWORDS and adds one more: *semantic*: the words semantically correlated must be close to each other in all word clouds. Several simulation studies as well as case studies are conducted to evaluate and demonstrate the effectiveness of the COWORDS algorithm.

Keywords: information retrieval, probabilistic text visualization, text visualization, semantic word cloud.

Sumário

Capítulo 1

Introdução

A análise exploratória de dados é uma etapa crucial na modelagem estatísitca. Existem diversas técnicas de análise exploratória e as suas aplicações variam de acordo com a natureza dos dados e também pela estrutura em que o dado é armazenado. Os gráficos pertecem ao conjunto de técnicas que compõem a análise exploratória e podem ser utilizados em todas as etapas de análise e modelagem. John W. Tukey foi um dos principais estatísticos responsáveis pelos avanços e também pela popularização da análise exploratória de dados. Tukey desenvolveu várias técnicas de visualização de dados, tais como, *box plot* e *stem and leaf*, os quais são amplamente utilizados. Em 1977, Tukey em seu livro, *Exploratory Data Analysis* define a análise exploratória de dados como:

> "Exploratory data analysis is detective work numerical detective work or counting detective work or graphical detective work. "

Esta definição deixa claro que o uso de ferramentas gráficas na exploração de dados tem um papel muito importante. Estas ferramentas permitem ao usuário uma melhor compreensão dos dados, facilitando por exemplo a detecção de padrões e também de anomalias, ou até mesmo confirmar alguma teoria do pesquisador. A importância da análise visual na exploração de dados motivou a formalização da área de Visualização de Dados, hoje também conhecida como Visualização da Informação. Edward R. Tufte foi um dos principais responsáveis pela formalização e crescimento da área com a sua coleção de livros publicados, e em destaque o livro *The Visual Display of Quantitative Information* [14]. Além de Tufte, outro autor que contribuiu estudando a fundo as diferentes percepções humanas para as mais variadas formas geométricas, cores e profundidade foi Jacques Bertin com o seu livro *Semiology of graphics: diagrams, networks, maps* [3]. Outro pesquisador que tem uma grande relevância em visualização de dados focado na exploração dos dados é William S. Cleveland com a publicação de dois livros *The elements of graphing data* em 1985 e *Visualizing data* em 1993 [6; 5].

As teorias desenvolvidas e formalizadas no passado servem de base para a pesquisa em visualização de dados. Além disso, o avanço tecnológico dos computadores juntamente com a

computação gráfica contribuiram para o desenvolvimento de ferramentas que impulsionaram o desenvolvimento de gráficos para análise de dados. Com estes avanços, não só dados de natureza numérica podem ser facilmente armazenados e analisados, como também documentos textos. Nesta tese, iremos focar na direção de visualizar documentos texto ao longo do tempo ou agrupados, no sentido de ajudar o analista a detectar possíveis padrões temporais nos conteúdos.

1 Visualização de dados

Existem algumas definições para *visualização de dados* na literatura [4; 3]. Por exemplo, Munzner [10] define como: "computer-based visualization systems provide visual representations of datasets designed to help people carry out tasks more effectively.". Enquanto a definição de Card et al. [4] é: "The use of computer-generated, interactive, visual representations of data to amplify cognition". Em outras palavras, visualização de dados é o processo de tranformar um conjunto de dados em uma representação visual facilitando o entendimento e a extração de informação dos dados.

Uma das principais vantagens da visualização de dados é descobrir padrões que não são esperados a partir de uma hipótese. Além disso, nenhuma suposição é necessária para aplicação das técnicas de visualização de dados. Naturalmente toda análise é guiada por alguma hipótese do pesquisador, caso contrário o analista precisaria fazer um grande número de combinações de análises.

O clássico exemplo de descoberta de padrões não esperados são os dados artificiais de Anscombe, chamado por Quarteto de Anscombe (*Anscombe's quartet*) o qual consiste de quatro conjuntos de dados. Curiosamente, Anscombe foi cunhado de Tukey, suas esposas eram irmãs, isto acabou contribuindo para que Anscombe e Tukey trabalhassem em parceria em Princeton no campo de análise e visualização de dados. Estes conjuntos de dados foram propostos por Anscombe em 1973 [1] para motivar o uso de gráficos estátisticos para revelar conhecimento que não são observados utilizando apenas estatísticas descritivas. Na Tabela 1.1 apresentamos os quatro conjuntos de dados com as suas respectivas variáveis. Algumas estatísticas descritivas foram calculadas para cada um dos conjuntos de dados e apresentadas na Tabela 1.2. As estatísticas descritivas apresentadas são iguais em todos os conjuntos de dados, até mesmo para os parâmetros da regressão linear. Se olharmos apenas para esta tabela somos induzidos a concluir que os quatro conjuntos de dados são praticamente os mesmos. Entretanto isto não é verdade. Na Figura 1.1 apresentamos o *scatterplot* para os quatro conjuntos de dados. Observamos que os conjuntos são bem diferentes, apesar das estatísticas descritivas serem exatamente as mesmas. Com esse conjunto de dados, Anscombe demonstra a importância da visualização de dados na análise exploratória. Outros conjuntos de dados com esta mesma intenção já foram proposto por outros autores, por exemplo veja o trabalho de Matejka e Fitzmaurice [9].

I		II		III		IV	
x	y	x	y	x	y	x	y
10,0	8,0	10,0	9,1	10,0	7,5	8,0	6,6
8,0	7,0	8,0	8,1	8,0	6,8	8,0	5,8
13,0	7,6	13,0	8,7	13,0	12,7	8,0	7,7
9,0	8,8	9,0	8,8	9,0	7,1	8,0	8,8
11,0	8,3	11,0	9,3	11,0	7,8	8,0	8,5
14,0	10,0	14,0	8,1	14,0	8,8	8,0	7,0
6,0	7,2	6,0	6,1	6,0	6,1	8,0	5,2
4,0	4,3	4,0	3,1	4,0	5,4	19,0	12,5
12,0	10,8	12,0	9,1	12,0	8,2	8,0	5,6
7,0	4,8	7,0	7,3	7,0	6,4	8,0	7,9
5,0	5,7	5,0	4,7	5,0	5,7	8,0	6,9

Tabela 1.1: Quarteto de Anscombe.

Descritivas	Valores
Média x	9,0
Variância x	10,0
Média y	7,50
Variância y	3,75
Correlação entre x e y	0,898
Regressão Linear	$y = 0.5x + 3.0$

Tabela 1.2: Estatísitcas descritivas para o Quarteto de Anscombe.

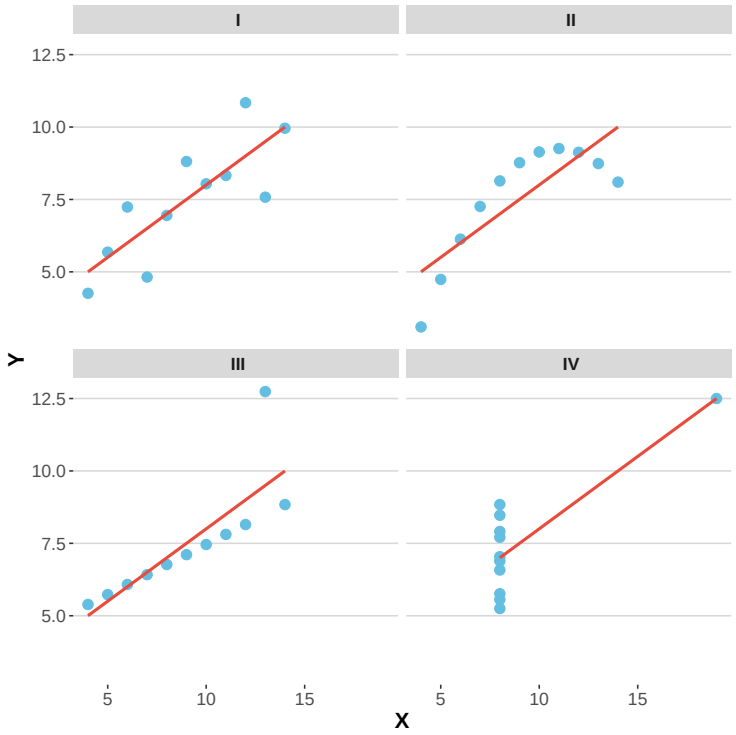


Figura 1.1: Scatterplot para cada conjunto de dados do Quarteto de Anscombe e suas respectivas reta de regressão.

Visualização de dados fornece ao usuário uma forma rápida, simples e intuitiva para compreensão dos dados. A sua aplicação ocorre nas mais diversas áreas de estudos, como biologia, física e ciências econômicas. Outra área de aplicação da análise exploratória visual é a de mineração de textos em que os conteúdos são extraídos e analisados no intuito de identificar os tópicos abordados nos documentos, bem como os sentimentos transmitidos pelos textos.

1.1 *Word Clouds*

Em visualização de textos, as nuvens de palavras se apresentam como uma das técnicas estatísticas mais efetivas na sumarização de textos em forma visual. A nuvem de palavras é uma representação gráfica do texto em que o tamanho da palavra é proporcional a sua importância no texto. A importância das palavras são normalmente mensuradas através da sua frequência no texto. Após reescalar o tamanho das palavras, estas são organizadas de forma a minimizar os espaços em branco entre as palavras, mas sem que ocorram sobreposições entre elas. O resultado desta organização das palavras lembra um formato de nuvem, daí o nome do gráfico. Na Figura 1.2 apresentamos uma típica nuvem de palavras gerada através do aplicativo Wordle [13] para o texto do livro *The Adventures of Sherlock Holmes*.

Figura 1.2: Exemplo de uma nuvem de palavras tradicional.

As nuvens de palavras tem uma grande popularidade pelo seu poder analítico e também por ser um gráfico visualmente agradável e de fácil interpretação. Além disso, existem diversos aplicativos para geração de nuvem de palavras que permitem aos usuários configurar diversas características das nuvens, por exemplo, os ângulos das palavras, número de palavras na nuvem e paleta de cores. O mais popular destes aplicativos é o Worlde [13] que introduziu o algoritmo espiral para o posicionamento das palavras de forma mais compacta possível.

Esta grande popularidade das nuvens de palavras fizeram com que alguns autores estudassem a sua efetividade como uma ferramenta de análise de dados [11; 2; 12; 8; 7]. Estes trabalhos corroboraram que as nuvens de palavras sejam muito eficazes em resumir e transmitir os conteúdos dos textos, um vez que permitem uma rápida idenficação dos principais tópicos abordados no texto, além de serem visualmente mais agradáveis aos usuários quando

comparadas com tabelas e listas de palavras.

Apesar da popularidade e da sua eficácia em ajudar o analista na identificação de tópicos, as nuvens de palavras tem algumas limitações quando são utilizadas para comparar diferentes documentos. Em geral, quando as nuvens de palavras são geradas, elas posicionam as palavras em ordem aleatória ou de acordo com a importância da palavra. Esta estratégia dificulta a comparação de dois ou mais documentos. Como as nuvens são geradas de forma independente uma das outras, as palavras que são compartilhadas nos diferentes documentos são posicionadas em regiões diferentes em cada nuvem, dificultando a análise ao longo das nuvens. Outra limitação das nuvens de palavras é fato de posicionar as palavras sem levar em consideração sua correlação com as outras. O ideal seria que as palavras que são posicionadas próximas uma das outras tivessem alguma correlação semântica. Por exemplo, palavras que aparecem várias vezes nas mesmas frases do texto deveriam ficar próximas umas das outras na nuvem de palavras.

Nesta tese abordamos estas limitações das nuvens de palavras utilizando uma distribuição de probabilidade para as nuvens. Esta distribuição de probabilidade tem alta densidade naquelas configurações de nuvens que permitem a sua comparação ao longo do tempo ou por grupos. Para isso, nós propomos o algoritmo Metropolis-Hastings para amostrar desta distribuição e então selecionar configurações com a maior densidade. Consequentemente, terminamos com uma configuração que procura solucionar as limitações.

1.2 Organização da tese

A tese está organizada de acordo com o conceito de coleção de artigos. Portanto, no Capítulo 2 apresentamos o algoritmo COWORDS que aborda o problema de comparar múltiplas nuvens de palavras. Este algoritmo produz nuvens em que as palavras compartilhadas ao longo do tempo permanecem na mesma posição. No Capítulo 3, apresentamos uma generalização do modelo probabilístico proposto no primeiro trabalho. Esta generalização permite criar múltiplas nuvens de palavras que, além de garantir que palavras compartilhadas fiquem na mesma posição em diferentes nuvens, ela também faz com que palavras correlacionadas permaneçam próximas uma das outras em todas as nuvens. Ambos trabalhos contribuem para o desenvolvimento da visualização de dados propondo duas novas formas de visualização de documentos textos, além de levar em conta um modelo probabílistico extremamente flexível que permite uma fácil extensão.

Referências

[1] Francis J Anscombe. Graphs in statistical analysis. *The American Statistician*, 27(1): 17–21, 1973.

[2] Scott Bateman, Carl Gutwin, and Miguel Nacenta. Seeing things in the clouds: the

effect of visual features on tag cloud selections. In *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*, pages 193–202. ACM, 2008.

[3] Jacques Bertin. Semiology of graphics: diagrams, networks, maps. 1983.

[4] Stuart K Card, Jock D Mackinlay, and Ben Shneiderman. *Readings in information visualization: using vision to think*. Morgan Kaufmann, 1999.

[5] William S Cleveland. *Visualizing data*. Hobart Press, 1993.

[6] William S Cleveland and William S Cleveland. *The elements of graphing data*. Wadsworth Advanced Books and Software Monterey, CA, 1985.

[7] Martin J Halvey and Mark T Keane. An assessment of tag presentation techniques. In *Proceedings of the 16th international conference on World Wide Web*, pages 1313–1314. ACM, 2007.

[8] Byron YL Kuo, Thomas Hentrich, Benjamin M Good, and Mark D Wilkinson. Tag clouds for summarizing web search results. In *Proceedings of the 16th international conference on World Wide Web*, pages 1203–1204. ACM, 2007.

[9] Justin Matejka and George Fitzmaurice. Same stats, different graphs: Generating datasets with varied appearance and identical statistics through simulated annealing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, pages 1290–1294. ACM, 2017.

[10] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[11] Anna W Rivadeneira, Daniel M Gruen, Michael J Muller, and David R Millen. Getting our head in the clouds: toward evaluation studies of tagclouds. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 995–998. ACM, 2007.

[12] James Sinclair and Michael Cardew-Hall. The folksonomy tag cloud: when is it useful? *Journal of Information Science*, 34(1):15–29, 2008.

[13] Julie Steele and Noah Iliinsky. *Beautiful Visualization: Looking at Data Through the Eyes of Experts*. O'Reilly Media, Inc., 1st edition, 2010. ISBN 1449379869, 9781449379865.

[14] Edward R Tufte and Glenn M Schmieg. The visual display of quantitative information. *American Journal of Physics*, 53(11):1117–1118, 1985.

Chapter 2

COWORDS: A Probabilistic Model for Multiple Word Clouds

Luís G. Silva e Silva and Renato M. Assunção

Abstract

Word clouds constitute one of the most popular statistical tools for visual analysis of text document because they provide users with a quick and intuitive understanding of the content. Despite their popularity for visualizing single documents, word clouds are not appropriate to compare different text documents. Independently generating word clouds for each document leads to configurations where the same word is typically located in widely different positions. This makes very difficult to compare more than two or more word clouds. This paper introduces COWORDS, a new stochastic algorithm to create multiple word clouds, one for each document. The shared words in multiple documents are placed in the same position in all clouds. Similar documents produce similar and compact clouds, making easier to compare and interpret simultaneously several word clouds. The algorithm is based on a probability distribution in which the most probable configurations are those with a desirable visual aspect, such as a low value for the total distance between the words in all clouds. The algorithm output is a set of word clouds randomly selected from this probability distribution. The selection procedure uses a Markov chain Monte Carlo simulation method. We present several examples illustrating the performance and the visual results that can be obtained by our algorithm.

Aceito para publicação ao *Journal of Applied Statistics* em 28 de outubro de 2017.

1 Introduction

Word cloud is one of the most effective statistical technique to summarize texts in a visual way. A word cloud consists of a graphical arrangement of the most frequent words of a given text document. These frequent words are scaled according to their frequency and tightly packaged without overlap forming a cloud shaped figure. Figure 2.1 shows a typical word cloud using the text of a presidential speech by Barack Obama. The word cloud is an approach widely used in data visualization, because it provides a way for people to grasp holistically the information, to explore, to summarize, and to understand the data. By simply glancing at the word cloud, we can figure out the main topics mentioned in the text and their relative importance.

The word cloud popularity is easily verified. A search on Google using the query "word cloud", retrieves approximately 150 million results. This popularity is partially explained by the existence of many word cloud generators freely available on the Internet. The most popular one is Wordle, which offers several options to create the word cloud such as the control of the words' angles and color palettes [21]. The algorithms behind such software are typically deterministic.

In textual data analysis, it is common the need to compare different documents with the aim to identify their similarities and to detect their differences. This comparison can be simply between two documents or among several document groups. For example, we can compare journal articles over time to study the variation of the most frequent topics or the emergence and disappearance of topics. The documents (the journal articles) are grouped by time interval and word clouds are built for each group. The clouds can then be compared in terms of similarities and differences. Another example is to compare the vocabulary use for different writers.

While word clouds are an effective way to visualize the content of one group of documents, it has one major disadvantage when they are used for comparison purposes between more than one group. Word clouds built for two or more groups may be substantially different in visual terms, even if the document groups are similar in content. Since the word clouds are run separately and independently for each group, the shared words are shown in different positions inside cloud. To visually compare the frequency change in a single word x present in two clouds, the user is forced to look for x within each cloud. To carry out this with many words simultaneously or with more than two clouds is a difficult and tiresome task. Even worse, it can be arduous and error prone to figure out that one word is present in one cloud but not in another one.

In order to illustrate the problem of comparing word clouds, we show in the top row of Figure 2.2 three word clouds built with the Wordle application. The texts are three Barack Obama speeches in different US presidential debates. The fifty most frequent words in each

www.wordle.net

speech are shown in the clouds. One can try to follow the evolution of a word across time. However, this is difficult because their size and position may change drastically. For example, the word "jobs" is clearly visible in the second speech due to its large size but it is harder to locate in the first and third speeches, when it has decreased substantially. Not only that, it has also moved from its conspicuous position to less visible locations. Not all words are so hard: "Romney" is more easily spotted across the clouds. Concerning "China", it takes some time for the user to notice that it shows up only in the third cloud. To reach this conclusion, the user should check every word in each cloud, almost an impossible task.

In this paper, we introduce a probabilistic model to build multiple word clouds. We develop an algorithm that searches for appropriate configuration of the words in the clouds. This search process is guided by three principles: (i) *tightness*: it requires that the returned configurations should have all clouds with a minimum empty space amount between the words; (ii) *overlapping*: the words in each cloud must have no overlap; (iii) *position*: the words must be in the same spatial location in each cloud where they appear. The two first principles are followed by all word cloud algorithms, either they consider one or more than one cloud. The third one is more difficult and there is only one previous work aiming at it, the deterministic *Word Storms* algorithm, described in Section 2. In the present paper, we adopt a different approach. Rather than outputting a unique configuration that optimizes a certain objective function as all the other current methods, we develop a probabilistic procedure. Our idea is to sample a cloud configuration from a probability distribution. The cloud sets with more probability mass are those satisfying our three guiding principles.

The bottom row of Figure 2.2 shows the result of our algorithm with the Obama speeches. The 50 most frequent words of each speech are shown in the form of a word cloud. If a word appears in different speeches, it is shown in the same position in all clouds. The word "jobs" is clearly spotted in South-Eastern region of the second cloud. It can be easily found in this same position in the first and third clouds, even though its size (or frequency) has decreased substantially. Words with more extreme dynamics can be easily identified by our method. "China", for example, appears in the top right position of the third cloud. Since the words positions are invariant in time in our method, the fact that we do not see "China" in this same top right location in the first and second clouds means that it is not a frequent word in these earlier speeches. In contrast with the Wordle algorithm run independently for each speech, the user does not have to spend time searching the clouds to check that "China" has disappeared. This example shows how our algorithm allows for direct comparison between different clouds. Our method follows all three guiding principles listed before. Due to the restriction imposed by the second principle, the tightness of each individual cloud in our method cannot be larger than that obtained running Wordle separately. However, it is surprising that the clouds tightness in the top and bottom rows are almost the same.

We built a Metropolis-Hastings algorithm, a special case of a Markov Chain Monte Carlo (MCMC) simulation method, to sample from the proposed clouds probability distribution. Our algorithm can easily incorporate additional constraints besides requiring the same po-

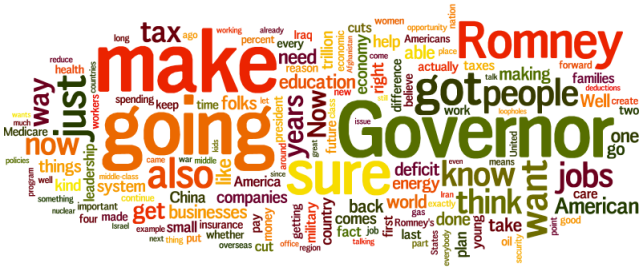


Figure 2.1: Example of a typical word cloud.

sition of the words in the different clouds. An additional aspect is the novelty of framing the search of a visually pleasing word cloud as a probabilistic problem.

In the next section, we present the related work in this problem. In Section 3, we present our probabilistic dynamic word cloud model, shortened to *COWORDS*, and our sampling procedure. In the next section, we present examples with simulated and real data showing the effectiveness of our method. Finally, in Section 5 presents a discussion and the conclusions.

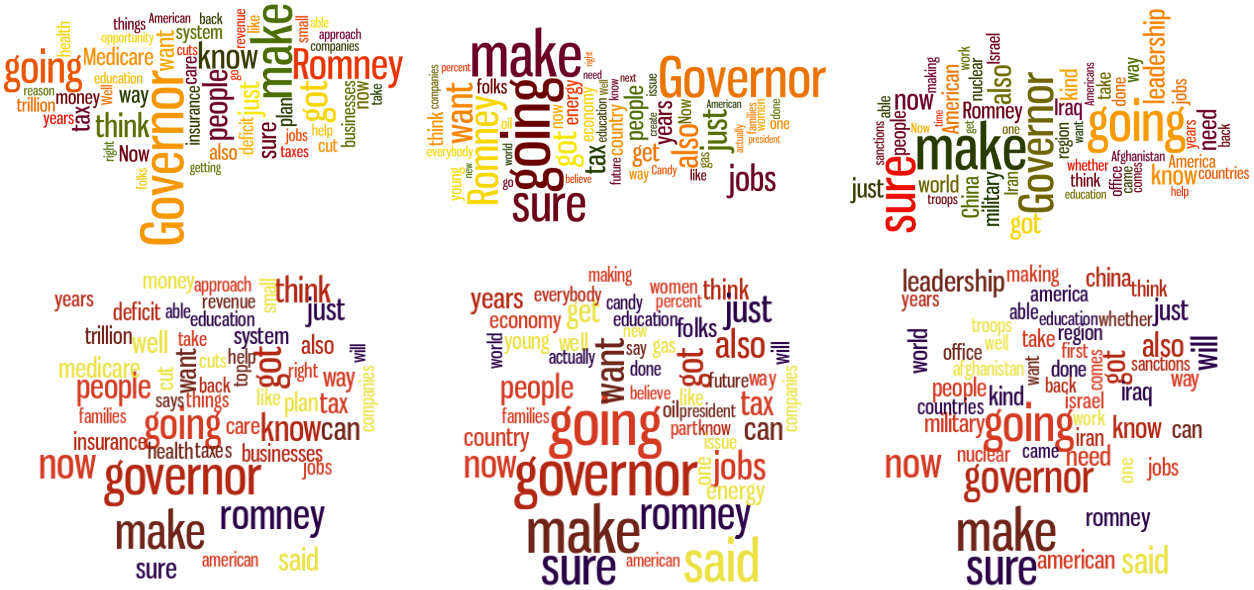


Figure 2.2: Comparison of different word clouds based on three Barack Obama presidential campaign speeches. Top row: word clouds built independently for each speech. Bottom row: word clouds built with our probabilistic algorithm, which constrains the words to stay in the same position in all clouds.

2 Related work

The word cloud popularity motivates some authors to study their efficiency with respect to other statistical summaries. One of them [11] compared the word clouds with ordered lists of most frequent words organized in a rectangular way. Participants were asked to locate and

select a certain word in each one of the two interfaces. The results showed that the terms were found more quickly using the ordered lists. The search time was strongly affected by the word font size and by the specific location of the words in the interfaces. Similar results were found by Kuo et. al. [13]. However, although the search time was faster for the ordered lists, the participants showed more satisfaction using the word clouds. In [20], the authors carried out an experiment where participants were exposed to some questions and they resort to either a word cloud or a more traditional table to help on finding the answers. The authors noted that the tables are preferred to answer specific questions while the word clouds are preferred when answering more general questions.

Other papers studied the effect of different word cloud characteristics in the recognition and posterior remembrance of specific words. Bateman et al. [3] and Rivadeneira et al. [16] found that properties such as the font size and weight have a larger impact than color and character number. They also found that words located on the cloud center are more easily recognized than others located on the cloud fringe. These works corroborate that word clouds are a useful tool to summarize and analyze the content of texts as they allow for the quick identification of their main topics and are visually pleasant when compared with statistical summaries organized as lists and tables.

All word cloud algorithms preprocess the text eliminating stop words, which are common words that do not express the text content, such as the words *the, a, is, are, which*. Optionally, words may also be stemmed by reducing similar words to one single radix (such as mapping the words *student, students, study, studied* to the single word *study*). After these steps, the frequency of the remaining words are calculated and a filter is applied to select the words to appear in the cloud. It can be either the k most frequent words or all the words that appear in the text more than k times. Finally, an algorithm is applied to arrange the words as a compactly shaped cloud.

The most popular algorithm to build a word cloud is that running on Wordle [21]. The first step of the algorithm estimates the area needed to display the set of words. This estimate is based on the sum of the bounding box area for each word. The words are sequentially placed according to their decreasing frequency or other relevant numerical weights. The i-word is randomly located around a horizontal center-line of the region cloud. If there is no intersection with the $i-1$ previous words already placed in the cloud, it is fixed at the trying position. Otherwise, it is moved in a spiral-like movement at fixed angle increments until it has no intersection with the other words. The intersection testing is made with a combination of hierarchical bounding boxes and quadtrees [24; 13]. Another common option is to change the placement sequence following a simple alphabetical order. ManiWordle [12] is an implementation that offers interactive features controlling the visual appearance generated by Wordle. For example, the user can interactively play with the positions, colors, and angles of each word in the cloud.

In the last years, several researchers worked with the problem of comparing word clouds. The Parallel Tag Cloud was presented by [7] in which the basic idea is to make a matrix

where the columns represent the time or groups and the rows represent the words. The words are put in each column in alphabetical order and the font size is proportional to its frequency. Common words between the columns are connected by a line to visualize their evolution. Although the display of the words in alphabetical lists is informative and of easy understanding, their work does not preserve the aesthetic of the word clouds rendering them not comparable.

Cui et al. [8] proposed a graphical way to show the historical evolution of document sets. The authors presented a new algorithm aiming at maintaining the semantically similar words next to each other in each one of the clouds.

There are other works with the objective of locating semantically similar words in the same region of the cloud. One of them is [25], which uses the seam carving technique to remove the empty spaces and to produce a more compact layout. The reference [2] is a comparison study of different techniques to create word clouds accounting for the words meaning.

Paulovich et al. [15] proposed ProjCloud to visualize and cluster texts. RadCloud [4] is a visualization technique that produces a single merged view of texts in different categories. As in the usual word cloud, the font size represents the word relevance. However, as the same word can appear in texts of different categories, the largest relevance value is selected to represent the word. Hence, it is not possible to track the word evolution and this is recommended only when one is interested in comparing only a few categories.

Other methods such as SparkClouds [14] and Document Cards [22] has the objective of adding new characteristics to the word clouds such as sparklines [23] and images. SparkClouds [14] combines the word cloud with sparklines to present the word frequency evolution. The Morphable Word Clouds [6] shows word clouds with different shapes. The objective is to present clouds in shapes that illustrate some dynamic aspect such as, for example, the different human life stages, from youth to old age.

Word Storm [5] is the first and, until now, the only attempt to solve the problem of visualizing simultaneously several word clouds. It outputs a group of clouds side by side, each one representing a single text document. Words that appear in multiple documents are placed in the same position, orientation and color in each one of the clouds. To obtain this effect, the authors propose two separate algorithms. In the first one, a solution is based on a trial and error method. It runs an algorithm similar to Wordle in each cloud independently. If the i-th word is shared by more than one cloud, it calculates the arithmetic average (i_x, i_y) of their spatial coordinates in the several clouds in which it is present. It loops over the shared words repositioning their centers in all clouds at the same position given by the mean (i_x, i_y). Typically, there will be a large amount of intersection between the words. Selecting some order, it runs the Wordle spiral-like movement in each cloud independently to spread the words and decrease overlap. Then, it iterates the procedure calculating again the arithmetic average (i_x, i_y) of each word, repositioning them and spreading with the spiral-like movement. This first algorithm may not converge especially if the number of clouds and

words is moderate or large.

The second algorithm uses an optimization approach. It defines a quadratic objective function that penalizes clouds in which the shared words appear in different positions. It uses a gradient descent method to minimize the objective function and the solution is the optimum minimum. The algorithm requires a very large number of iterations to converge and it strongly depends on the initial configuration of the words. It can also converge to a solution exhibiting words overlap in a cloud.

The problems faced by the authors of Word Storm motivated them to combine the two algorithms. The first one is used to generate an initial value for the second one. This combined method produced better results than the individual algorithms.

3 A Probabilistic Model for Multiple Word Clouds

The three word clouds shown in the top row of Figure 2.2 are redesigned in the bottom row to exhibit the most important property for dynamic clouds: each word holds the same position across different clouds. Let $w_i = (x_i, y_i)$ be the time-invariant center position of the i-th word and $\mathbf{W} = (w_1, w_2, \ldots, w_n)$. The Euclidean distance between the words i and j is $d_{ij} = |w_i - w_j|$. Since the word positions are the same for all t, these distances do not change in time and can be represented by a line segment $w_i w_j$ as shown in Figure 2.3(a). Let α_{tij} be the length of the subsegment connecting the intersection points of the segment $w_i w_j$ with the words' rectangles boundaries.

Although d_{ij} is constant in time, the value of α_{tij} can change substantially if the words change their frequency over time. In Figure 2.3(b), the j-th word increased substantially, while the i-th word decreased its frequency resulting in a larger $\alpha_{2ij} > \alpha_{1ij}$.

Following our guiding principles to generate tight clouds, we want α_{tij} as small as possible. Furthermore, they must be non-negative to avoid overlapping words. However, requiring $\alpha_{tij} > 0$ is not enough to guarantee non-overlapping words. The reason can be seen in the configuration of words l and m in the Southeast location in Figure 2.3(b). Although $\alpha_{2ml} > 0$ in this case, we have a non-empty intersection between the words' rectangles. Therefore, we need to impose a more restricted check to verify if a configuration is viable.

3.1 COWORDS algorithm

Rather than casting this problem as an optimization task, we establish a probabilistic formulation. We define a probability density for each possible cloud sequence configuration. The density puts more probability mass on those configurations that follow more closely our guiding principles. The more satisfactory as a solution, the higher the probability the configuration is selected. More specifically, we randomly select positions $\mathbf{W} = (w_1, w_2, \ldots, w_n)$

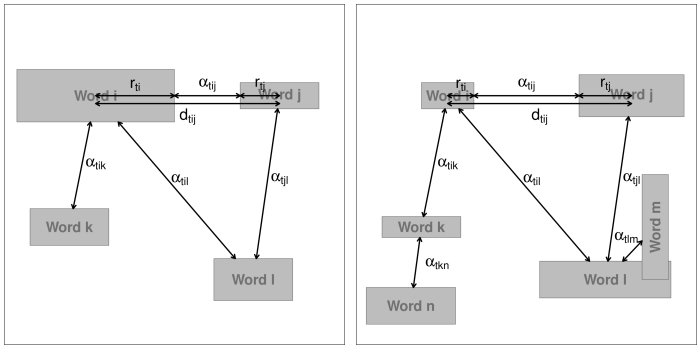
from the following probability density function:

$$\pi(\mathbf{W}) \propto \exp\left\{-\sum_{t=1}^{T}\sum_{i\sim j}\alpha_{tij}^2\right\}\prod_{t=1}^{T}\prod_{i\neq j}\mathbb{1}_{[S_{tij}]} \tag{2.1}$$

where $\mathbb{1}_{[S_{tij}]}$ is the indicator function

$$\mathbb{1}_{[S_{tij}]} = \begin{cases} 1, & \text{if there is no overlap between } i \text{ and } j \text{ at time } t. \\ 0, & \text{otherwise.} \end{cases} \tag{2.2}$$

The probability distribution support is over all configurations with no overlap between the word rectangles. On this set, the density proposed ensures that, when α_{tij} decreases, the density increases in order to satisfy the tightness principle. The exponential factor in the density (2.1) ensures a positive function, as it must in the case of probabilities, as well as it enforces a more stringent penalty for slack configurations. The density (2.1) is specified without a normalizing constant. This nasty constant will not be necessary, as we explain below.



(a) Time 1. (b) Time 2.

Figure 2.3: Words configuration example. The parameter α_{tij} represents the distance between words i and j at time t. The configuration in time 2 shows that we can have words overlapping even when all $\alpha_{tij} > 0$.

In Fig. 2.4, we compare two configurations with $T = 1$ that have non-zero density under (2.1). Both are possible to be sampled from (2.1), but the more compact configuration, on the right-hand side of the figure, has a density 214430.2 times higher than the sparse configuration on the left-hand side of the figure. To find configurations with the desired features, it is sufficient to sample a large number of configurations from the distribution (2.1) and then to select those with high probability density $\pi(\mathbf{W})$.

To sample the highly dependent variables $\mathbf{W} = (w_1, \ldots, w_n)$ according to the target distribution (2.1), we use a Markov Chain Monte Carlo (MCMC) algorithm [17]. We adopted a Gibbs sampler algorithm with a Metropolis-Hastings step. The main advantage of the Gibbs sampler is that it breaks down a multivariate simulation in a sequence of univariate simulations. To apply the Gibbs sampler, it is necessary to calculate the conditional distribution of each of the positions w_k conditional on all the others. Let $\mathbf{W}_{-k} =$

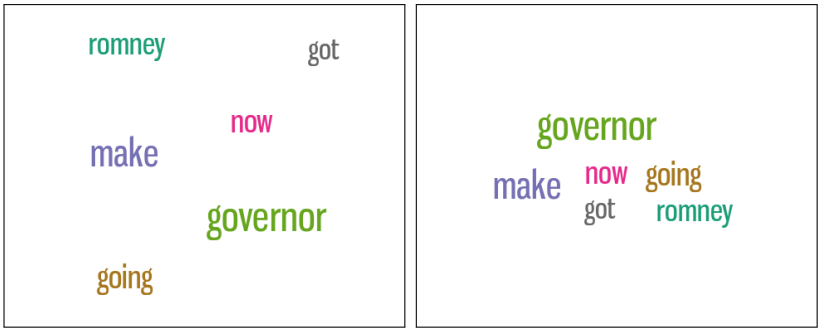


Figure 2.4: Two configurations that are possible according to the distribution (2.1). The cloud on the right-hand side has density 214430.2 times higher than the cloud on the left-hand side.

$(w_1, \ldots, w_{k-1}, w_{k+1}, \ldots, w_n)$ be the set of all positions except the k-th word position. We obtain the conditional distribution $\pi_c(w_k|\mathbf{W}_{-k})$ for each k-th word conditioned on the current positions $\mathbf{W}_{-k}$ of all other words by retaining from (2.1) only the multiplicative factors involving the variable w_k. That is:

$$\begin{aligned} \pi_c(w_k|\mathbf{W}_{-k}) &\propto \pi(\mathbf{W}) \\ &\propto \exp\left\{-\sum_{t=1}^{T}\sum_{j:j\sim k}\alpha_{tkj}^2\right\}\prod_{t=1}^{T}\prod_{j:j\neq k}\mathbb{1}_{[S_{tkj}\geq 0]} \end{aligned} \tag{2.3}$$

Only j and t are varying within the exponent and the product in (2.3). The index k is held constant and equal to the k-th word for which we are calculating the conditional distribution. The normalizing constant of the full conditional (2.3) does not have a closed form and hence it is not possible to sample directly from this distribution. As a consequence, we use the Metropolis-Hastings step to generate from this full conditional probability density. The idea of the Metropolis-Hastings algorithm is to use an auxiliary and simpler distribution, called proposal distribution, from which we know how to sample directly. The proposed value typically depends on the current configuration making the successive random draws stochastically dependent. Given a value generated from this proposal distribution, a test is performed to accept or reject it as a value generated from our target distribution (2.3). Given that this proposal distribution is arbitrary and possibly very different from the full conditional (2.3), the test mixes both probability distributions used to ensure that we are generating values from (2.3).

At step m and word k, we draw the proposed value w^* from a multivariate Gaussian distribution centered at the current k-th word position $w_k^{(m-1)}$ and with covariance matrix $\mathbf{\Sigma} = \sigma^2\boldsymbol{I}$. Let $\mathcal{N}(w^*; w_k^{(m-1)}, \mathbf{\Sigma})$ be the bivariate Gaussian density at w^* value and

$$\rho\left(w_k^{(m-1)}, w^*\right) = \min\left\{1, \frac{\pi_c\left(w^*|\mathbf{W}_{-k}\right)\mathcal{N}\left(w_k^{(m-1)}; w^*, \mathbf{\Sigma}\right)}{\pi_c\left(w_k^{(m-1)}|\mathbf{W}_{-k}\right)\mathcal{N}\left(w^*; w_k^{(m-1)}, \mathbf{\Sigma}\right)}\right\}.$$

We then accept w^* as the new value $w_k^{(m)}$ with probability $\rho(w_k^{(m-1)}, w^*)$. If it is not accepted,

we maintain $w_k^{(m)} = w_k^{(m-1)}$. The Gaussian property of symmetry around the mean allows us to simplify $\rho\left(w_k^{(m-1)}, w^*\right)$ by canceling out the $\mathcal{N}$ density factor in both, numerator and denominator as they are equal for this specific choice of proposal distribution. The complete algorithm is in shown as Algorithm 1 below.

Algorithm 1 COWORDS algorithm

Require: Number of iterations M and optionally an initial configuration.
Ensure: Final position of words $\mathbf{W} = \{w_k\}_{k\in\{1,\ldots,n\}}$.

if starting position not set **then**
  for all $k \in \{1, \ldots, n\}$ **do**
    Generate $\mathbf{w}_k^{(0)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  end for
end if
for all $m \in \{1, \ldots, M\}$ **do**
  for all $k \in \{1, \ldots, n\}$ **do**
    $w^* \sim \mathcal{N}_2(w_k^{(m-1)}, \boldsymbol{\Sigma})$
    Take $\rho\left(w_k^{(m-1)}, w^*\right) = \min\left\{\frac{\pi_c(w^*)}{\pi_c\left(w_k^{(m-1)}\right)}, 1\right\}$.
    Generate $u \sim \mathcal{U}(0, 1)$
    if $u < \rho\left(w_k^{(m-1)}, w^*\right)$ **then**
      $w_k^{(m)} \leftarrow w^*$
    else
      $w_k^{(m)} \leftarrow w_k^{(m-1)}$
    end if
  end for
end for

Although any choice of σ^2 is theoretically valid for the Metropolis-Hastings algorithm, in practice it is the most critical aspect of this algorithm. For one side, a small value of σ^2 will produce a high acceptance rate but a slow mixing chain. That is, a restricted search space for the proposed configuration. For the other side, a large variance will produce a small acceptance rate because most of time an unsuitable configuration is proposed. A good choice of σ should induce an average acceptance rate around 0.234 [10].

3.2 Bubble chart algorithm: from words to symbols

There are situations when we want to visualize a dynamic table of values in the form of a symbol cloud rather than a word cloud. As an example, consider the premier league table of different soccer seasons in Brazil. Figure 2.5 shows the first ten best teams in each year, from 2009 to 2014. The team shields are proportional to the championship points accrued in that year. We can adapt our multiple word cloud to generate the symbol cloud seen in Figure 2.5. In fact, the algorithm is simpler than COWORDS due to the possibility of using circles, rather than rectangles, to represent the cloud items.

In each time $t \in \{1, 2, \ldots, T\}$, we have the same set of n circles. The Euclidean distance between the centers of circles i and j in time $t \in \{1, 2, \ldots, T\}$ is given by d_{tij}. The radii of the circles i and j at time t is represented by r_{ti} and r_{tj} and it is proportional to a numerical value. We are interested in minimizing the distances $\alpha_{tij} = d_{tij} - (r_{ti} + r_{tj})$ between the boundaries of the circles for the entire period (see Figure 2.6).

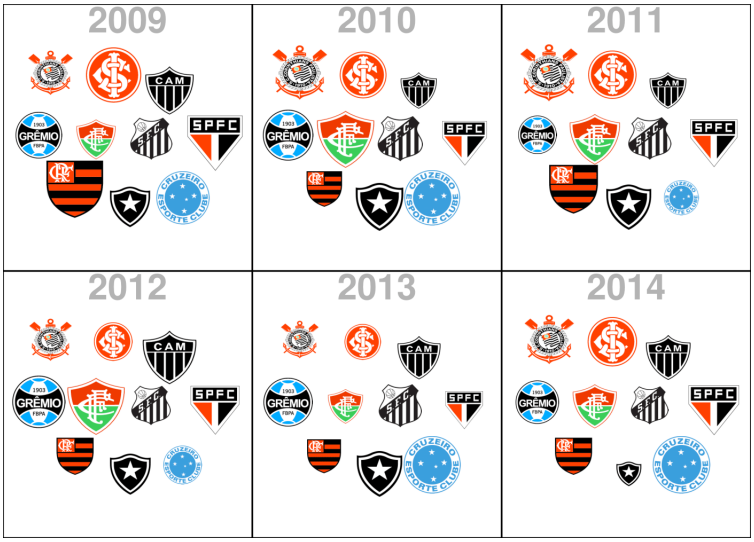


Figure 2.5: The 10 best teams in the premier Brazilian soccer tournament in each year, from 2009 to 2014. The team shields are proportional to the points accrued in that year.

In contrast with the COWORDS algorithm, when $\alpha_{tij} \geq 0 \; \forall \; t,i$ and j we have a configuration in which there is no overlapping among circles. This is a much simpler test to evaluate if a proposed configuration is valid than in the word-rectangle case. In the latter, the relative positions and orientations of the rectangles must be taken into account to calculate α_{tij}, as we have discussed.

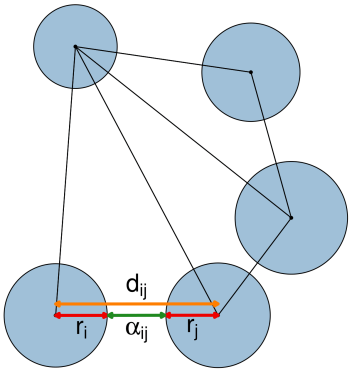


Figure 2.6: Bubble chart algorithm: using circles to enclose symbols rather than rectangles and words.

The bubble chart algorithm changes the probability distribution and the α_{tij} definition:

$$\pi(\mathbf{W}) \propto \exp\left\{-\sum_{t=1}^{T}\sum_{i\sim j}\alpha_{tij}^2\right\}\prod_{t=1}^{T}\prod_{i\neq j}\mathbb{1}_{[\alpha_{tij}\geq 0]} \tag{2.4}$$

with

$$\mathbb{1}_{[\alpha_{tij}\geq 0]} = \begin{cases} 1 & \alpha_{tij} \geq 0 \\ 0 & \alpha_{tij} < 0 \end{cases} \tag{2.5}$$

where the notation $i \sim j$ shows that the circle i is linked to circle j being, therefore, neighbors. The probability distribution proposed ensures that when α_{tij} increases, the density

decreases, and for negative values of α_{tij} the density is zero. Note that the part of the model that is inside the shaded rectangle in (2.4) is the one that ensures compact clouds, while the part inside the shaded ellipse ensures that there is no overlap. Figure 2.5 is the output of the bubble chart algorithm.

3.3 Further details

Presently, COWORDS utilizes an initial configuration chosen completely at random. That is, we select the n positions $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ independently and following a uniform distribution within a large rectangle. Typically, there is much overlap among the initial rectangles enclosing the words. However, quickly COWORDS find configurations satisfying the non-intersecting words principle. See details about the algorithm convergence in Section 4.

One alternative for the initial configuration is to run the usual Wordle algorithm with the maximum size for each word. This will guarantee that there is no overlap between the enclosing rectangles for all times $t \in \{1, 2, \ldots, T\}$. However, we did not find much gain in using this approach as COWORDS quickly drift away from our random initial choice towards reasonable configurations.

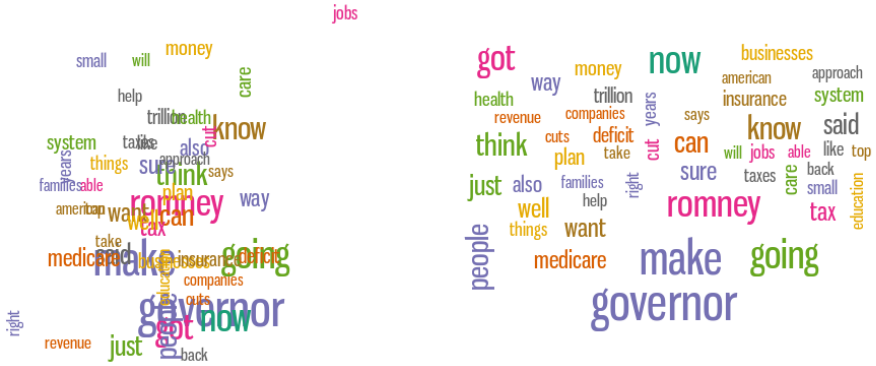
It is important to notice that there are many different configurations $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ leading to local maxima of the probability density $\pi(\mathbf{W})$ in (2.1). Indeed, given any value for $\mathbf{W}$, all its rigidly rotated configurations, or any symmetrical mirror image $\mathbf{W}$ will have exactly the same density value. It is not difficult to see that different configurations will lead to local maxima for (2.1). In Section 4 we give examples showing that although our algorithm can output different final configurations, the probability density evaluated in each one of them is approximately the same. The relevant point is that there is no single objective maximum value associated with a single optimal configuration but rather many possible cloud configurations leading to approximately the same pleasant visual aspect satisfying the guiding principles of Section 1.

4 Experiments

In this section, we illustrate the COWORDS algorithm and assess its functioning with several examples. Initially, we show properties of COWORDS, such as the effect of the initial configuration and its convergence speed, using a single word cloud and multiple clouds. Next, we show COWORDS in action in many applications highlighting its output aspects rather than the algorithm characteristics. In this section, we used the notation $\boldsymbol{\alpha}$ to represent the amount $\sum_{t=1}^{T} \sum_{i \sim j} \alpha_{tij}^2$.

4.1 Algorithm characteristics: single cloud

Consider the single word cloud composed by the $n = 50$ most frequent words of the first Barack Obama speech, shown in the first plot of the top row (also shown in the first plot of the bottom row) in Figure 2.2. To illustrate the quickly disappearing effect of the initial configuration, we start with the cloud showed in Figure 2.7(a). The outcome of the algorithm after $M = 100,000$ iterations is in Figure 2.7(b). We can see that even with the large number of words and the large amount of overlapping among them, the algorithm proved to be very efficient: it removed all overlapping and provided a compact word cloud. It is also noticeable that the final positions of the words are very different from their initial positions. We measured the amount of these displacements between the initial and final configurations. Imagine a coordinate system with the origin in the center of the initial cloud configuration and the resulting four quadrants. We find that 74% of the words' centroids changed quadrants between the initial and the final configurations, showing that most of the words ended up away from where it started.



(a) Algorithm Initial configuration. (b) Algorithm Final configuration.

Figure 2.7: Simulation study with 50 words and $T = 1$ using the most frequent words of Barack Obama first speech, in Oct 13, 2012.

As the $M = 100,000$ iterations were performed, COWORDS search for configurations attending our principle guides. At the 444-th iteration, the algorithm begins to return non-overlapping configurations. Therefore, few iterations after starting with a bad configuration, our algorithm was able to solve the overlapping problem while maintaining compactness. As the proposed algorithm belongs to the class of acceptance-rejection algorithms, it is recommended to analyze its acceptance rate. The acceptance rate is the total number of proposals accepted in the Metropolis step divided by the total number of iterations. The acceptance rate is a measure to assess the quality of the sampler. For instance, if the acceptance rate is too high, then the variance used in the proposal distribution, $\mathcal{N}_2(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, is too small. In this case, the algorithm will take longer than necessary to find the target distribution. If the acceptance rate is too low, the proposal variance is too large and the algorithm is inef-

fective in the exploration of values space. In the worst case no value is accepted and so the algorithm fails to move. [18; 19] study the optimal acceptance rate for some models under specific conditions finding 23% as a good value.

In this first example, the global acceptance rate was 20.1%. In Figure 2.8, we represent the global acceptance rate with a horizontal red line. This is a good rate since the model has a high dimension. Another way to assess our algorithm acceptance rate is to calculate the acceptance rate of subsets of the iterations. These subsets are called windows and we use J to represent the length of the window. For instance, with a window length equal to $J = 100$, we calculate the acceptance rate for the first 100 samples, then for the next 100, and so on. There is no overlapping between windows. In Figure 2.8, we present the acceptance rate for windows of length $J = 100$, implying in 1000 sequential time windows. Note that the acceptance rate is high in the first windows. This result is to be expected, since the initial configuration is very bad and therefore we need to update the positions of the words in 75% of the first 100 iterations. In Table 2.1 we present the acceptance rates in the first 6 windows. The rates remain higher than the overall 20.1% rate in all windows displayed. We note in Figure 2.8 that as the number of iterations increase, there is a natural decrease in the acceptance rate. This algorithm behavior occurs because, after finding a good configuration, the words positions will be hardly updated.

Windows	1	2	3	4	5	6
Rate	75%	45%	46%	29%	43%	29%

Table 2.1: Acceptance rate for windows of length 100.

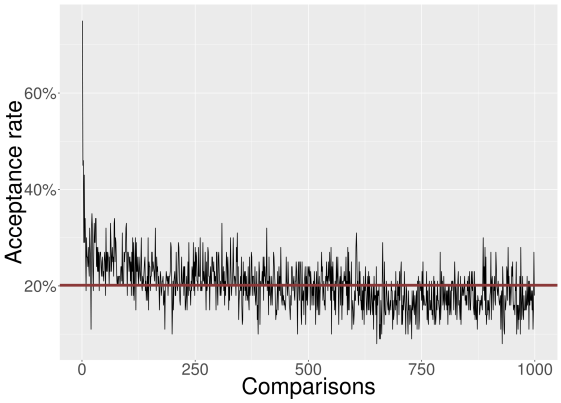


Figure 2.8: Acceptance rate for window of length 100 along time.

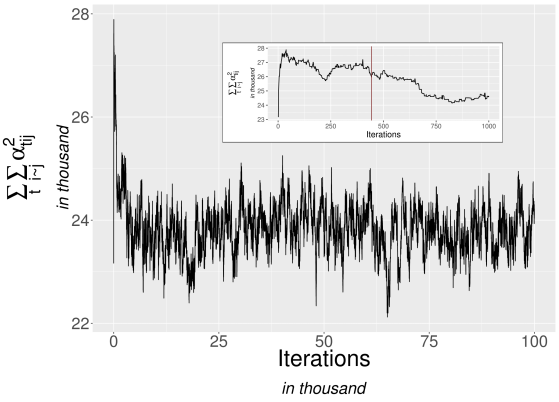
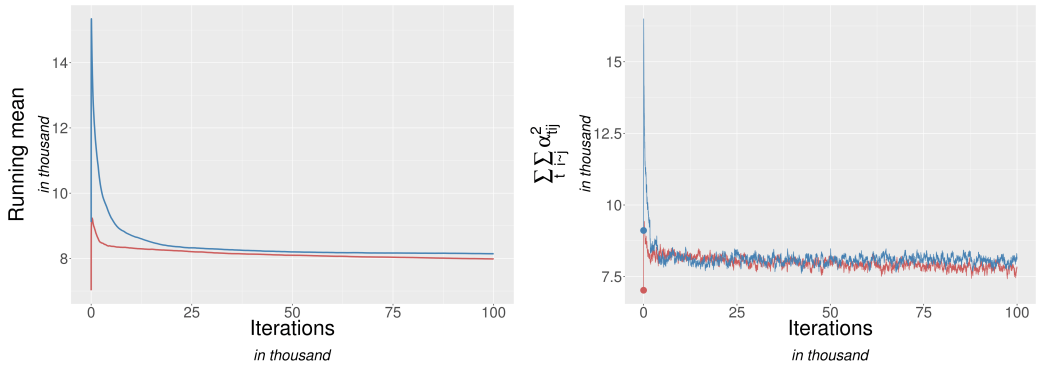


Figure 2.9: Trace plot of $\boldsymbol{\alpha}$ and a subplot of the 1000 first iterations.

In Figure 2.9, we present a trace plot of the amount $\boldsymbol{\alpha}$ for each one of the 100 thousand configurations. COWORDS aim is to sample configurations with probability proportional to $\exp(-\boldsymbol{\alpha})$ and hence stimulating the appearance of configurations with low values for $\boldsymbol{\alpha}$ but with no overlapping words. Indeed, Figure 2.9 shows a decreasing general trend of this parameter. However, in the subplot, we can observe that there is a fast increase of $\boldsymbol{\alpha}$ in the first iterations while it runs away from configurations with overlapped words. The

vertical red line at the subplot highlights the iteration of number 444, since when there is no overlapping among the circles. After this initial phase is overcome, there is a systematic general decreasing trend along the iterations meaning that the algorithm is searching for configurations that minimize $\boldsymbol{\alpha}$ while respecting the non-overlapping condition. As $\boldsymbol{\alpha}$ is a function of the Markov chain generated by our algorithm, we can use it to evaluate the algorithm convergence. In Figure 2.9, we can see that the chain of $\boldsymbol{\alpha}$ tends to stabilize around a certain value. This chain behavior suggests that convergence was achieved. We can also see that this is a strongly correlated chain, with values changing slightly from one iteration to the next one.

A more detailed analysis of convergence, although visual, is shown in Figure 2.10. It presents two chains of $\boldsymbol{\alpha}$ generated by our algorithm with different initial configurations. In Figure 2.10(a) we show the time series of the running mean of $\boldsymbol{\alpha}$ for both chains. The running mean at iteration k is computed as the mean of all $\boldsymbol{\alpha}$ sampled values up to and including k. A time series of the running mean of the chain allow us to check whether the chain is slowly or quickly approaching its target distribution. The black horizontal line with the mean of the chain facilitates this comparison. When there is convergence, the expected output is a curve approaching a horizontal line at a fixed value, the stationary distribution mean. In addition, we expect that independently run chains should converge to the same mean. In fact, this expected output can be verified in Figure 2.10(a), where both chains are approaching the same mean although they started from very different initial configurations. Therefore, this provides evidence for the chain convergence. We also present the trace plot of $\boldsymbol{\alpha}$ for the two chains, which is an essential plot for assessing and diagnosing Markov chain convergence. It basically shows the time series of $\boldsymbol{\alpha}$ and its most desirable outcome is a kind of "white noise" plot where the two chains should show a good amount of mixing on their traces. In the Figure 2.10(b), we can see that both chains are mixing quite well along the time and are approximately "white noise". So we have a strong indication of the algorithm convergence.



(a) Running mean plot for $\boldsymbol{\alpha}$.

(b) Traceplot for two chains with different initial points.

Figure 2.10: Convergence diagnostics of the $\boldsymbol{\alpha}$.

4.2 Algorithm characteristics: multiple clouds

The second experiment assesses how the algorithm performs when there are many simultaneous word clouds, the main objective of our algorithm. In particular, we are interested in seeing how the algorithm behaves when we have drastic changes in the words' frequencies over time. We considered the lyrics from all songs in eight Beatles' albums spread over their career. The eight albums were divided into the four first albums (Please, please me; With the Beatles; A Hard Day's Night; Beatles for Sale) and the four last ones (Yellow Submarine; The Beatles ; Let it Be; Abbey Road). We used the 35 most frequent words in the collected songs of each album.

Figure 2.11: Most frequent words in song lyrics from eight albums released by The Beatles.

In Figure 2.11, we present the output of our algorithm. We ran it for 100 thousand iterations and selected that with the minimum value of α. There are no overlapping words and they hold the same position along time. The algorithm was also able to keep a small distance between the contiguous words even in the presence of large temporal size variation. For instance, the frequency of words "love", "yeah", and "honey" has changed drastically across time. However, COWORDS successfully found a configuration with the desired property. This allows us to compare simultaneously a large number of word clouds such as the eight clouds in this example. We can notice, for example, that the first four clouds have a small number of words with high frequency with most words presenting a small size. The last two albums, in contrast, show a much more balanced frequency of the words. This could mean a richer set of themes for the last songs or a more diverse interest reflected in the lyrics. Another explanation is that the first songs style often had short and repetitive chorus, emphasizing few words such as "love" and "yeah". It is striking the small frequency of the word "love" in the albums "The Beatles" and "Let it Be", when the relationship between the band members was stressed and conflict among them was common. Supposedly, the climate in the last album improved and this was reflected in the larger size of that word.

The acceptance rate for each window of size $J = 100$ is presented in Figure 2.12. It took 129 iterations for the algorithm to solve the overlapping problem using a random starting configuration. The global acceptance rate is around 1%, which is very low. The acceptance rate for the first period of 100 iterations was 83%, whereas for the next five periods were 55%, 30% 23%, 17% and 13% respectively. Despite the quick acceptance rate decrease, the value of $\boldsymbol{\alpha}$ also decreases quickly. In Figure 2.13 we show the evolution of $\boldsymbol{\alpha}$. This plot shows that we could stop much earlier, such as at the 25000-*th* iteration, and still obtain a reasonably compact set of word clouds.

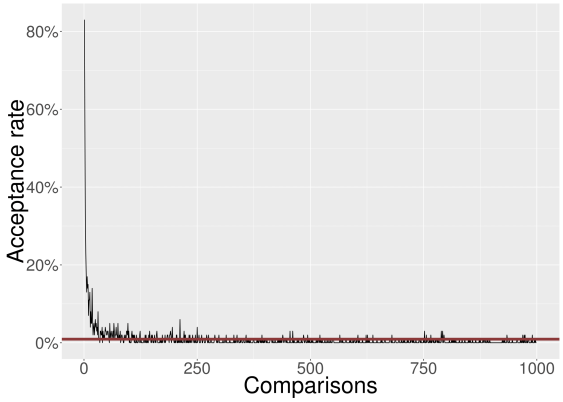


Figure 2.12: Acceptance rate for window of size 100.

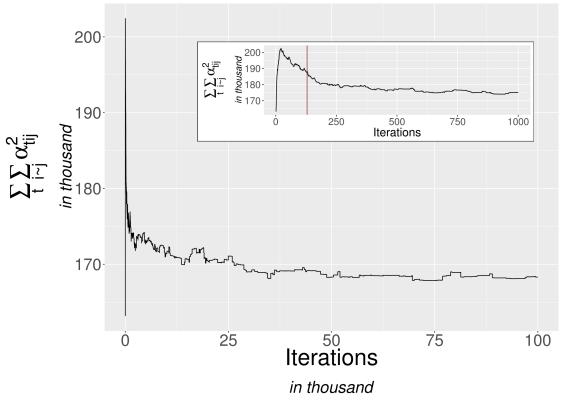
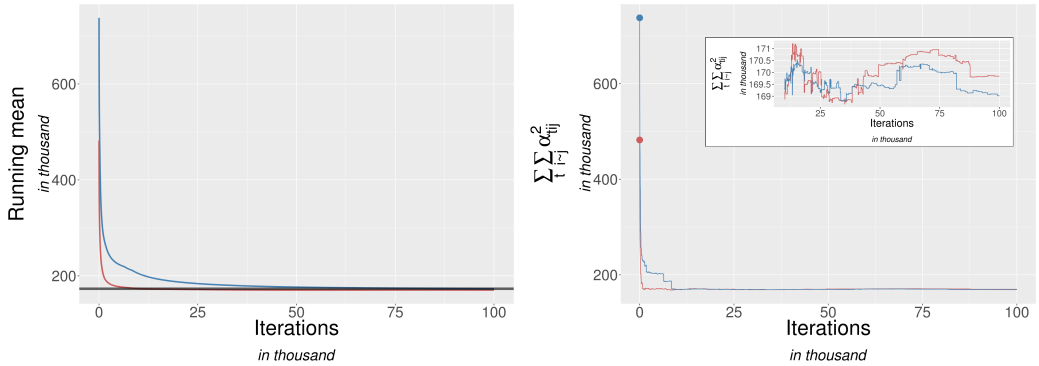


Figure 2.13: Trace plot for $\boldsymbol{\alpha}$ and a subplot of the 1000 first iterations.

To analyze the convergence of the algorithm, we present again the plots of the running mean in Figure 2.14(a) and the trace plot for two chains from $\boldsymbol{\alpha}$ with different initial points in Figure 2.14(b). The former shows that both chains are approaching the same average as the iterations increase. The trace plot in Figure 2.14(b) shows that the two chains are mixing along the time although it is not as good as in the case with only one word cloud, as we see in the inset. In the present case, both chains go to the same value range but they have a strong serial correlation between successive values which makes them vary slowly in the state parameter. This does not compromise the quality of the final clouds, it only says that one may need to run the algorithm for a large number of iterations (such as the 100K times we used here) in order to obtain a satisfactory output.

4.3 The federalist papers

In this section, we illustrate COWORDS with the federalist papers, a collection of 85 newspaper articles written between 1787 and 1790 by Alexander Hamilton, James Madison, and John Jay, American revolutionaries involved in the United States independence. We selected only 80 of them, divided into four groups: 51 authored by Hamilton, 15 by Madison, 3 jointly written by Hamilton and Madison, and 11 with authorship disputed between Hamilton and Madison. Figure 2.15 shows the 35 most frequent words in each group of texts as a result of running COWORDS for 100 thousand iterations. It is clear that, while the two authors do not have remarkable differences in their use of the most frequent words



(a) Running mean plot for $\boldsymbol{\alpha}$.

(b) Traceplot for two chains with different initial points.

Figure 2.14: Convergence diagnostics of the $\boldsymbol{\alpha}$ for the Beatles example.

when writing alone, they used quite different ones when writing together. The disputed set of articles does not appear clearly favoring either of them. The simple word frequency is not enough to distinguish between the two writers.

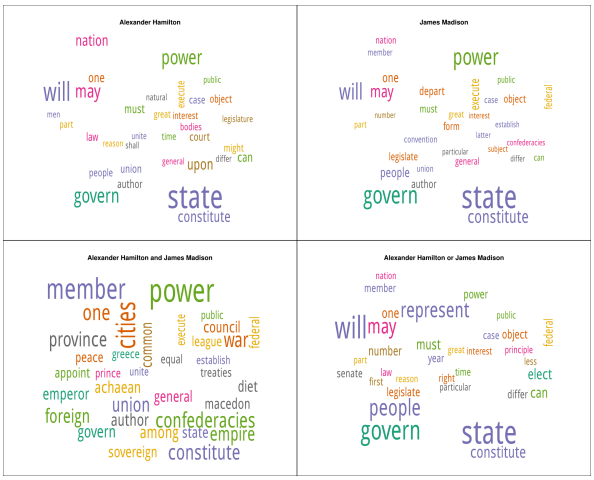


Figure 2.15: Most frequent words of the federalist papers divided into four groups: those authored only by Hamilton, only by Madison, by both, and of disputed authorship.

4.4 Brazilian movies

In this application, we consider the titles of 3,670 movies produced in Brazil from 1908 to 2015 [9]. We classified the movies in decades and ran our COWORDS algorithm. Figure 2.16 shows the result. The video in the supplemental material is richer as it allows to appreciate in a dynamic way how the movie themes evolved in time. There was a clear trend in the interests and topics. In the first decades, movies were typically light and naive comedies presenting merry widows ("viúva alegre") visiting "Rio" de Janeiro and meeting counts ("conde") during the carnival ("carnaval"). Starting in the 60's, social themes ("sertão", "violência", "morte") dispute with erotic topics ("mulheres", "virgem", "sexo") until the amazing dominance of the sexual thematic in the eighties. From 1990 on, a more diverse set of subjects divide the

attention of the public and no single topic is easily discernible.

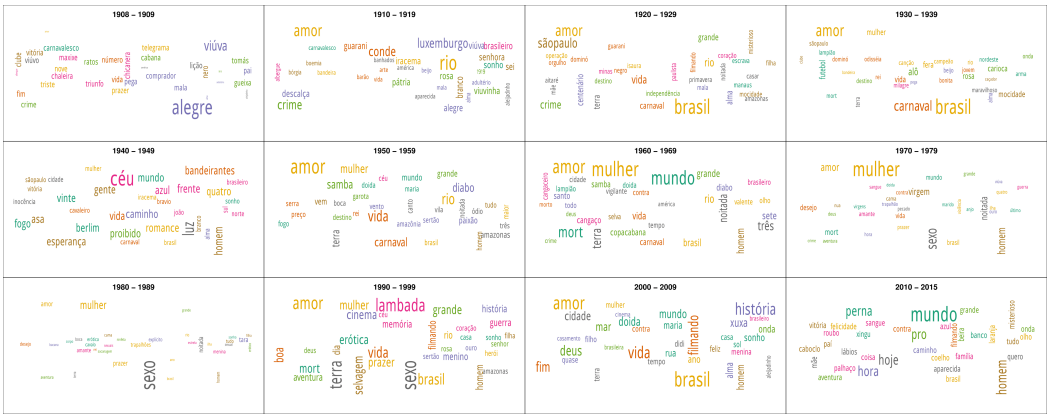


Figure 2.16: Most frequent words in the titles of 3670 Brazilian movies produced between 1908 and 2015.

4.5 Comparison with Word Storm

We compared COWORDS with Word Storm, the only alternative method to generate multiple word clouds, to the best of our knowledge. We used the Obama speech example, shown in the lower row of Figure 2.2. In order to compare both algorithms, we used the *tightness* metrics [1]. It measures how tight is the word cloud and it is one of the three principles for building clouds of words described on Section 1. Ideally, we want to minimize the total empty space between words in all clouds to avoid wasted space in the visual display. The tightness metrics is defined as $\delta = 1 - (\text{used area})/(\text{total area})$, where the used area is the sum of the areas of all the rectangles that wrap each word. For the total area, we consider two possibilities. The first one is the bounding box area containing all rectangles wrapping the words, while the second one is the convex hull involving all rectangles. The smaller the δ, the better the result. Table 2.2 shows the δ computed for both algorithms using different number of words: 50, 75, and 100. We used the Word Storm implementation freely available at the GitHub web site maintained by the authors' method. Qualitatively, the word clouds produced by Word Storm were visually similar to those produced by COWORDS. Quantitatively, the methods also produced similar results, with no method being uniformly better than the other. Although small, the largest differences occurred at times 2 and 3 when considering the convex hull as the total area. At time 2, COWORDS left less empty space than Word Storm, while the reverse happens at time 3. Considering the bounding box for the total area, our algorithm is better than Word Storm at time 2. In the other times, they are similar. Hence, from an visual efficiency point of view, the algorithms produce similar results. However, as pointed out by a reviewer, the main appeal of COWORDS is the great flexibility provided by the probabilistic approach of COWORDS. We discuss these differences with Word Storm in Section 5.

		Convex hull			Bounding box		
	Words	Time 1	Time 2	Time 3	Time 1	Time 2	Time 3
COWORDS	50	38,3%	**27,6%**	**38,4%**	43,2%	**29,8%**	45,5%
	75	38,3%	31,2%	37,8%	49,2%	41,7%	48,5%
	100	42,2%	**31,8%**	**41,9%**	47,8%	**38,7%**	48,8%
Word Storm	50	38,2%	**35,6%**	**31,0%**	49,6%	**52,1%**	48,3%
	75	41,8%	33,1%	34,3%	51,6%	46,9%	47,8%
	100	42,6%	**38,1%**	**32,2%**	50,8%	**51,5%**	45,4%

Table 2.2: *Tightness* metrics for both algorithms COWORDS and Word Storm by different number of words, 50, 75, and 100 words.

5 Conclusion

The main advantage of the word cloud text document representation is its immediate interpretation, in addition, to be a visually pleasing representation. The semiotic aspect of the word clouds is one of its main appeals: the word is self-represented and its size shows its frequency. The symbol represents an object and, at the same time, is that same object. Its popularity is great and it is currently used everywhere. Despite this popularity, the simple comparison of two word clouds is hard to do. It can be virtually impossible if we want to look simultaneously at four or more of them.

As pointed by one reviewer, one of the main advantages of our method compared to Word Storm is the easy incorporation of constraints or additional considerations to build the clouds. Rather than imposing hard restrictions on shape, position or other visual aspects and then carrying out a constrained minimization, we can simply define a different energy function in (2.1) by incorporating soft restrictions and simulate from the new distribution. This allows the introduction of additional constraints which can be dealt with relatively easily within the MCMC framework and opens the possibility of exploring a more structured way to organize multiple word clouds.

For instance, we can envisage dynamic word clouds with a label imposing contiguity restrictions in each moment of time. Imagine that words are classified into exclusive categories, such as “positive” or “negative” for sentiment analysis. We can impose that same category words should be close to each other and in the same position at all times. As our model is flexible, we can modify the neighborhood structure to accommodate this restriction. Another topic of future work is the visualization of a text collection with a large number of time points. An easy solutions is to aggregate the word clouds that are similar, leaving the user with fewer clouds to analyze.

In summary, in this paper, we presented a novel probabilistic model to build word clouds and symbol clouds. It is especially useful if one desires to compare multiple word clouds such as word clouds for multiple times or word clouds for different categories of texts. Our COWORDS algorithm places the words in the same location across clouds minimizing the space between them.

References

[1] L. Barth, S.G. Kobourov, and S. Pupyrev, *An experimental study of algorithms for semantics-preserving word cloud layout* .

[2] L. Barth, S.G. Kobourov, and S. Pupyrev, *Experimental comparison of semantic word clouds*, in *International Symposium on Experimental Algorithms*. Springer, 2014, pp. 247–258.

[3] S. Bateman, C. Gutwin, and M. Nacenta, *Seeing things in the clouds: the effect of visual features on tag cloud selections*, in *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. ACM, 2008, pp. 193–202.

[4] M. Burch, S. Lohmann, F. Beck, N. Rodriguez, L. Di Silvestro, and D. Weiskopf, *RadCloud: Visualizing multiple texts with merged word clouds*, in *Information Visualisation (IV), 2014 18th International Conference on*. IEEE, 2014, pp. 108–113.

[5] Q. Castellà and C. Sutton, *Word Storms: Multiples of Word Clouds for Visual Comparison of Documents*, in *Proceedings of the 23rd International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee, WWW '14, 2014, pp. 665–676. Available at http://dx.doi.org/10.1145/2566486.2567977.

[6] M.T. Chi, S.S. Lin, S.Y. Chen, C.H. Lin, and T.Y. Lee, *Morphable word clouds for time-varying text data visualization*, IEEE transactions on visualization and computer graphics 21 (2015), pp. 1415–1426.

[7] C. Collins, F. Viegas, and M. Wattenberg, *Parallel Tag Clouds to explore and analyze faceted text corpora*, in *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, Oct. 2009, pp. 91–98.

[8] W. Cui, Y. Wu, S. Liu, F. Wei, M.X. Zhou, and H. Qu, *Context preserving dynamic word cloud visualization*, in *Pacific Visualization Symposium (PacificVis), 2010 IEEE*. IEEE, 2010, pp. 121–128.

[9] A.L. da Silva Neto, *Dicionário de Filmes Brasileiros*, Antônio Leão da Silva Neto, São Paulo, SP, Brazil, 2002.

[10] J.S.R. Gareth O. Roberts, *Optimal scaling for various metropolis-hastings algorithms*, Statistical Science 16 (2001), pp. 351–367. Available at http://www.jstor.org/stable/3182776.

[11] M.J. Halvey and M.T. Keane, *An assessment of tag presentation techniques*, in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1313–1314.

[12] K. Koh, B. Lee, B. Kim, and J. Seo, *Maniwordle: Providing flexible control over wordle*, IEEE Transactions on Visualization and Computer Graphics 16 (2010), pp. 1190–1197.

[13] B.Y. Kuo, T. Hentrich, B.M. Good, and M.D. Wilkinson, *Tag clouds for summarizing web search results*, in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1203–1204.

[14] B. Lee, N.H. Riche, A.K. Karlson, and S. Carpendale, *Sparkclouds: Visualizing trends in tag clouds*, Visualization and Computer Graphics, IEEE Transactions on 16 (2010), pp. 1182–1189.

[15] F.V. Paulovich, F. Toledo, G.P. Telles, R. Minghim, and L.G. Nonato, *Semantic wordification of document collections*, in *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2012, pp. 1145–1153.

[16] A.W. Rivadeneira, D.M. Gruen, M.J. Muller, and D.R. Millen, *Getting our head in the clouds: toward evaluation studies of tagclouds*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 995–998.

[17] C.P. Robert and G. Casella, *Monte Carlo Statistical Methods (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[18] G.O. Roberts, A. Gelman, and W.R. Gilks, *Weak convergence and optimal scaling of random walk Metropolis algorithms*, Annals of Applied Probability 7 (1997), pp. 110–120. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.6598.

[19] G.O. Roberts and J.S. Rosenthal, *Optimal scaling for various metropolis-hastings algorithms*, Statist. Sci. 16 (2001), pp. 351–367. Available at http://dx.doi.org/10.1214/ss/1015346320.

[20] J. Sinclair and M. Cardew-Hall, *The folksonomy tag cloud: when is it useful?*, Journal of Information Science 34 (2008), pp. 15–29.

[21] J. Steele and N. Iliinsky, *Beautiful Visualization: Looking at Data Through the Eyes of Experts*, 1st ed., O'Reilly Media, Inc., 2010.

[22] H. Strobelt, D. Oelke, C. Rohrdantz, A. Stoffel, D.A. Keim, and O. Deussen, *Document cards: A top trumps visualization for documents*, IEEE Transactions on Visualization and Computer Graphics 15 (2009), pp. 1145–1152.

[23] E.R. Tufte, *Beautiful evidence* (2006).

[24] F. Viegas, M. Wattenberg, and J. Feinberg, *Participatory visualization with wordle*, Visualization and Computer Graphics, IEEE Transactions on 15 (2009), pp. 1137–1144.

[25] Y. Wu, T. Provan, F. Wei, S. Liu, and K.L. Ma, *Semantic-preserving word clouds by seam carving*, in *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2011, pp. 741–750.

Chapter 3

A probabilistic model for multiple word clouds preserving semantic correlation

Luís G. Silva e Silva and Renato M. Assunção

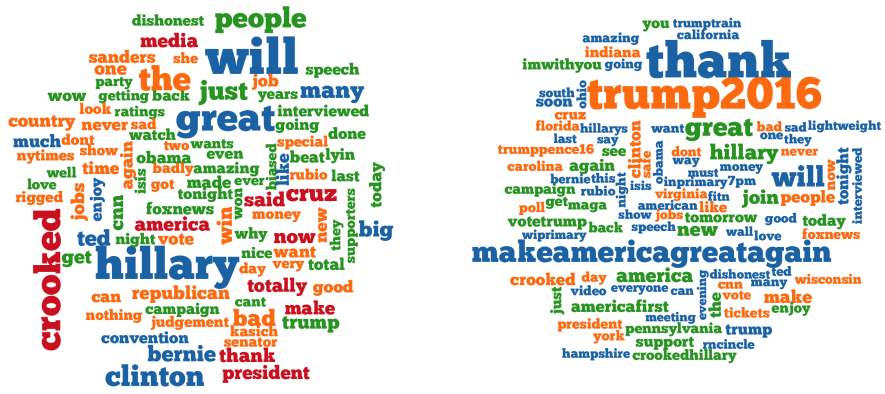
Abstract

Word clouds are the most popular statistical graphic to effectively summarize the content of a text document. They provide a quick and intuitive understanding of the content. Some word cloud generators incorporate their semantic content putting related words close to each other in a single cloud. Other algorithms attempt to solve the problem of creating multiple related word clouds such as temporally indexed clouds or clouds for different topics. The main objective of this class of algorithms is to facilitate the comparison of multiple word clouds. This paper introduces a new probabilistic algorithm that combines the two problems. Each word configuration has a certain probability that is concentrated on those patterns with the desired visual aspect: the same word should have the same position across multiple clouds and semantically words should be close to each other. The algorithm samples from this probability distribution over the word clouds configuration. The sampling procedure is based on a Markov chain Monte Carlo approach. We present several examples illustrating the performance and the visual results that can be obtained with our algorithm.

1 Introduction

Word cloud is one of the most effective statistical tool to visually summarize textual documents. It consists of a graphical arrangement of the most frequent words appearing in

a text or in a collection of texts assembled into a single text document. Figure 3.1 depicts a typical word cloud created based on the content of Donald Trump's tweets on the period from December 2015 to August 2016 sent from an Android device (left) and from an IPhone (right). The words are scaled according to their frequency and tightly packed, without overlapping, forming a cloud-shaped output. Word clouds are widely used in data visualization because they let people to holistically grasp the information contained in the text and they also provide a fast way to explore, summarize and understand the data. A quick glance at the word cloud allows us to figure out the main topics covered in the text and their relative importance.



(a) (b)

Figure 3.1: Most frequent words of Donald Trump's tweets from different devices on the period from Dec. 2015 to Aug. 2016. Left: sent from Android. Right: sent from IPhone.

Comparing documents is a commonplace task in textual data analysis. The main goal is to detect both similarities and differences among the documents. Not rarely, this comparison needs to be conducted for several groups of documents. For example, collecting all articles published in a given time period by a newspaper into a single large document, the objective is to track how the news content changed as time passes. One word cloud for each time period would be generated and their comparison should lead to clues about the dynamically changing news topics. Figure 3.1 shows another possibility, where two documents need to be compared with the texts separated out according to a categorization (a device type, in this case). The IPhone cloud, on the right hand side, has a more intense use of hashtags, such as *#makeamericagreatagain* and *#trump2016*. More polite words, such as *thanks* has a higher frequency (and hence, size) in the right hand side IPhone cloud than the left hand side Android cloud. In the latter, we have more struggling and aggressive tweets, frequently citing his opponent Hillary Clinton and using words such as *crooked*.

However, while word clouds are an effective way to visually explore a single group of documents, comparing several groups using word clouds is not an easy and straightforward task. A major drawback, in this case, is that the visual aspect of clouds built for two or more

groups of documents may substantially differ. Once the clouds are created in a separate and independent way for each group, the words they share can be placed in completely different positions in each cloud. To track the frequency (importance) change of a single word w across different clouds, the user needs to undertake an exhaustive search over every cloud looking for the word w without previous knowledge of whether w is present in all clouds. This task can easily become infeasible as the number of words and clouds grows large. Each word cloud in Figure 3.1 was created independently and they illustrate this difficulty.

In addition to this problem, another potential shortcoming of traditional word clouds is that they do not consider potential correlations between words in any way, as the word location is completely independent of their context. However, some empirical studies [8], [18] indicate that humans tend to correlate words that are close to each other. This issue has motivated the emergence of several new algorithms which attempt to compute layouts where semantically related words are closely placed [7; 29; 10; 18; 15; 28; 1]. All these studies approached the problem of the semantic word cloud from the static point of view and neither of them considers semantic word clouds over time.

In this paper, we propose a new probabilistic algorithm that hold common words in the same position in different clouds and additionally constrain the output configuration in a way that words near each other are semantically related. The algorithm has four principles that guide the search for desirable cloud configurations: (i) *tightness*: the final configurations of all clouds should present a small amount of empty space between the words; (ii) *overlapping*: the words in each cloud must have no overlap; (iii) *position*: the words must lie in same spatial location in each of the clouds that they appear; (iv) *semantic*: the words semantically correlated must be close to each other in all word clouds. The first two principles are followed for all word cloud algorithms in the literature. The third principle and fourth are more complex and usually not taken into account by most approaches. The fourth principle guides some word cloud algorithms (see Section 2 for further details), however, they do not take into account the third and fourth principles together. The third principle was considered previously by [4], that proposed WordStorm, and by our own previous paper [19], that proposed COWORDS.

Our present paper builds on COWORDS, allowing it to generate multiple word clouds preserving the semantic position of the words across all clouds. Our idea consists of incorporating a parameter that modifies the probability distribution to concentrate the probability mass in word cloud configurations that preserve the context of the words, in addition to holding the common words in the same position across different clouds. In this way, we can sample a cloud configuration from that probability distribution and, with high probability, outputs clouds with the desired visual features, those satisfying the four guiding principles. The probability distribution is multidimensional and complicated because the words' positions are highly correlated to each other. Hence, it is not simple how to generate samples from the proposed probability distribution. We employ a Metropolis-Hastings algorithm, that is a special case of a Markov Chain Monte Carlo (MCMC) simulation method, to

sample from the proposed clouds probability distribution. Additionally to considering the semantic content of the words, we considered other proposal distribution for the MCMC algorithm different from those used by COWORDS.

The remaining of the paper is organized as follows. In Section 2, we present a literature review on algorithms for word clouds. Section 3 review the COWORDS algorithm and present a new proposal distribution as alternative to the Gaussian distribution. In Section 4, we describe our proposal to generalize the COWORDS algorithm to account for the semantic content of the words. In Sections 4.2 and 4.3, we present experimental results using simulated and real-world data to illustrate the effectiveness of our method. Finally, Section 5 presents some discussion and concluding remarks.

2 Related work

The word cloud popularity motivates some authors to study their efficiency to convey the text content with respect to other statistical summaries. In [9], the authors compared the word clouds with ordered lists of the most frequent words organized in a rectangular way. Participants were asked to locate and select a certain word in each one of the two interfaces. The results showed that the terms were found more quickly using the ordered lists. The search time was strongly affected by the word font size and by the specific location of the words in the interfaces. Similar results were found by [12]. However, although the search time was faster for the ordered lists, the participants showed more satisfaction using the word clouds. In [20], the authors carried out an experiment where participants were exposed to some questions and they resort to either a word cloud or a more traditional table to help on finding the answers. The authors noted that the tables are preferred to answer specific questions while the word clouds are preferred when answering more general questions.

Other papers studied the effect of different word cloud characteristics in the recognition and posterior recollection of specific words. Bateman et al. [2] and Rivadeneira et al. [16] found that properties such as the font size and weight have a larger impact than color and character number. They also found that words located on the cloud center are more easily recognized than others located on the cloud fringe. These works corroborate that word clouds are a useful tool to summarize and analyze the content of texts as they allow for the quick identification of their main topics and are visually pleasant when compared with statistical summaries organized as lists and tables.

All word cloud algorithms preprocess the text eliminating stop words, which are common words that do not express the text content, such as the words *the, a, is, are, which.* Optionally, words may also be stemmed by reducing similar words to one single radix (such as mapping the words *student, students, study, studied* to the single word *study*). After these steps, the frequency of the remaining words are calculated and a filter is applied to select the words to appear in the cloud. It can be either the k most frequent words or all the words

that appear in the text more than k times. Finally, an algorithm is applied to arrange the words as a compactly shaped cloud.

The most popular algorithm to build a word cloud is Wordle [23]. The first step of the algorithm estimates the area needed to display the set of words. This estimate is based on the sum of the bounding box area for each word. The words are sequentially placed according to their decreasing frequency or other relevant numerical weights. The ith-word is randomly located around a horizontal center-line of the cloud region. If there is no intersection with the $i-1$ previous words already placed in the cloud, it is fixed at the putative position. Otherwise, the ith-word is moved in a spiral-like movement at fixed angle increments until it has no intersection with other words. The intersection test is made through a combination of hierarchical bounding boxes and quadtrees [26; 13]. Another common option is to change the placement sequence following a simple alphabetical order. ManiWordle [11] is an implementation that offers interactive features controlling the visual appearance generated by Wordle. For example, the user can interactively play with the positions, colors, and angles of each word in the cloud.

Temporal Word Clouds

In the last years, several researchers worked with the problem of generating and comparing multiple word clouds indexed by a variable such as a time stamp or a category. The Parallel Tag Cloud was introduced by [6]. Rather than generating the usual elliptical cloud, its this basic idea is to make piles of words and connect them with line segments. Its basic idea is to make a matrix where the columns represent the different index values (such as the different times or groups) and the rows represent the words. The words are put in each column in alphabetical order and the font size is proportional to its frequency. Common words between the columns are connected by a line to visualize their evolution. Although this connected linear display of words in alphabetical lists is informative and of easy understanding, their work does not preserve the aesthetic of the word clouds rendering the techniques not directly comparable.

Other methods such as SparkClouds [13] and Document Cards [24] have the objective of adding new characteristics to the word clouds such as sparklines [25] and images. SparkClouds [13] combines the word cloud with sparklines to present the word frequency evolution. The Morphable Word Clouds [5] shows word clouds with different shapes. The objective is to present clouds in shapes that illustrate some dynamic aspect such as, the different human life stages, from youth to old age.

Word Storm [4] is the first and, until now, the only attempt to solve the problem of visualizing simultaneously several word clouds. It outputs a group of clouds side by side, each one representing a single text document. Words that appear in multiple documents are placed in the same position, orientation and color in each one of the clouds. To obtain this effect, the authors propose two separate algorithms. In the first one, a solution is based on a

trial and error method. It runs an algorithm similar to Wordle in each cloud independently. If the i-th word is shared by more than one cloud, it calculates the arithmetic average (i_x, i_y) of their spatial coordinates in the several clouds in which it is present. It loops over the shared words repositioning their centers in all clouds at the same position given by the mean (i_x, i_y). Typically, there will be a large amount of intersection between the words. Selecting some order, it runs the Wordle spiral-like movement in each cloud independently to spread the words and decrease overlap. Then, it iterates the procedure calculating again the arithmetic average (i_x, i_y) of each word, repositioning them and spreading with the spiral-like movement. This first algorithm may not converge especially if the number of clouds and words is moderate or large. The second algorithm uses an optimization approach. It defines a quadratic objective function that penalizes clouds in which the shared words appear in different positions. It uses a gradient descent method to minimize the objective function and the solution is the optimum minimum. The algorithm requires a very large number of iterations to converge and it strongly depends on the initial configuration of the words. It can also converge to a solution exhibiting words overlap in a cloud. The problems faced by the authors of Word Storm motivate them to combine the two algorithms. The first one is used to generate an initial value for the second one. This combined method produced better results than the individual algorithms.

Semantic Word Clouds

While several studies tackled the problem of analyzing text documents over time and producing aesthetic visualizations, their layout algorithm typically do not incorporate the neighborhood relationships between words. That is, they do not place semantically related words close to each other in the word cloud. This is one of the critical limitations of traditional word cloud methods, as mentioned by Hearst [10]. In order to overcome this issue, Wu et al. [28] and Xu et al. [29] proposed similar approaches. Wu et al. [28] compute a distance matrix between words. They then use multidimensional scaling to place the words onto a 2D canvas and to remove empty spaces aiming at producing a more compact layout. Xu et al. [29] employ the same framework of Wu et al. [28] with slightly differences in the way they compute the similarity among the words. Xu et al. [29] similarity is based on word embeddings, such as word2vec [14]. Paulovich et al. [15] proposed ProjCloud to visualize and cluster texts. RadCloud [3] is a visualization technique that produces a single merged view of texts in different categories. As in the usual word cloud, the font size represents the word relevance. However, as the same word can appear in the content of different categories, the largest relevance value is selected to represent the word. Hence, it is not possible to track the word evolution and it is recommended only when one is interested in comparing a small number of categories. Wang et al. [27] proposed a method for consistently editing word clouds called EdWordle. In a nutshell, the tool allows users to move and edit words putting similar words vicinity while preserving the word cloud structure.

As far as we know, Cui et al. [7] is the only one work that dealt with both problems, the multiple word clouds and the semantic proximity restriction. The clouds are created independently in each time period. In each word cloud, they have a similarity matrix between the words and this matrix is used with a multidimensional scaling algorithm to locate the words in a planar region. After this, they applied an additional algorithm to eliminate the words overlapping. Therefore, there is no restriction that the common words should keep the same position across clouds. The user will have difficulty comparing the text content in two different groups or time moments as the words will be in different positions. The intention of [7] is to monitor how much the words changed their relative positions. This will indicate a change on the topics or sentiments connected with the documents. They propose an entropy-based measure to track the dynamic evolution of the word clouds.

3 COWORDS algorithm

The proposed probabilistic model to generate word clouds with semantics builds on the COWORDS model proposed by Silva and Assunção [19]. This is a stochastic algorithm to create multiple word clouds in which the shared words in multiple documents are placed in the same position in all clouds. The algorithm is based on a probability distribution in which configurations with a desirable aspect have a larger probability of being chosen, while configurations that are far away from desirable aspect have a low probability. The algorithm output is a multiple word clouds configuration randomly selected from this probability distribution. The selection procedure uses a Markov chain Monte Carlo simulation method. More specifically, it randomly select positions $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ from the following probability density function:

$$\pi(\mathbf{W}) \propto \exp\left\{-\sum_{t=1}^{T}\sum_{i\sim j}\alpha_{tij}^2\right\}\prod_{t=1}^{T}\prod_{i\neq j}\mathbb{1}_{[S_{tij}]} \tag{3.1}$$

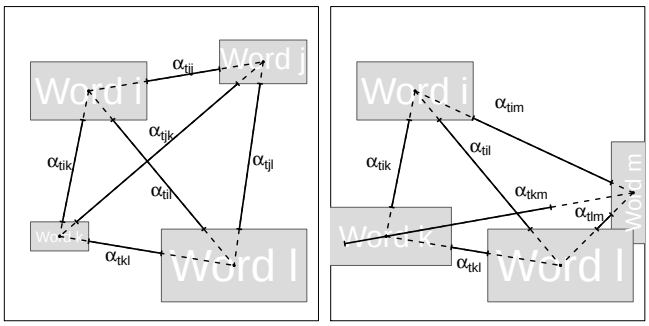
where $\mathbb{1}_{[S_{tij}]}$ is the indicator function.

$$\mathbb{1}_{[S_{tij}]} = \begin{cases} 1, & \text{if there is no overlap between } i \text{ and } j \text{ at time } t. \\ 0, & \text{otherwise.} \end{cases} \tag{3.2}$$

Support of the probability distribution is over all configurations with no overlap between the words. The exponential factor has the sum over all α_{tij}, the length of the connected edge that lies outside of rectangles' boundaries as shown in Figure 3.2(a) and $w_i = (x_i, y_i)$ is the time-invariant center position of the i-th word. Therefore, the Euclidean distance between the center position of the words i and j defined by $d_{ij} = |w_i - w_j|$ is also constant over time. In Figure 3.2(a), d_{ij} is represented by the line segment connecting the centers of words w_i and w_j.

The idea behind the probability density function (3.1) is that tight word clouds imply

in small values of α_{tij}. Therefore, it was built taking account that when α_{tij} decreases, the density increases in order to ensure compact clouds. In addition, the indicator function (3.2) ensures that for word cloud with overlapping between words have probability density equal to zero. The density (3.1) is specified without a normalizing constant. This nasty constant will not be necessary, as we explain below.



(a) Time 1 ($t = 1$). (b) Time 2 ($t = 2$).

Figure 3.2: Schematic illustration of two word clouds, for times $t = 1$ and $t = 2$. The parameter α_{tij} is the length of the connected edge between words i and j at time t that lies outside of the rectangles' boundaries.

Given that the probability distribution has the desired features of a word cloud, the next step of COWORD is to sample a large number of configurations from the distribution (3.1) and then to select those with high probability density $\pi(\mathbf{W})$. To sample from (3.1) it is used a Gibbs sampler algorithm with a Metropolis-Hastings step that belongs to Markov Chain Monte Carlo (MCMC) algorithm class [17]. The main advantage of the Gibbs sampler is that it breaks down a multivariate simulation in a sequence of univariate simulations. To apply the Gibbs sampler, it is necessary to calculate the conditional distribution for each of the positions w_k conditional on all the others. Let $\mathbf{W}_{-k} = (w_1, \ldots, w_{k-1}, w_{k+1}, \ldots, w_n)$ be the set of all positions except the k-th word position. The conditional distribution $\pi_c(w_k|\mathbf{W}_{-k})$ is obtained for each k-th word conditioned on the current positions $\mathbf{W}_{-k}$ of all other words by retaining from (3.1) only the multiplicative factors involving the variable w_k. That is:

$$\begin{aligned} \pi_c(w_k|\mathbf{W}_{-k}) &\propto \pi(\mathbf{W}) \\ &\propto \exp\left\{-\sum_{t=1}^{T}\sum_{j:j\neq k}\alpha_{tkj}^2\right\}\prod_{t=1}^{T}\prod_{j:j\neq k}\mathbb{1}_{[S_{tkj}\geq 0]} \end{aligned} \tag{3.3}$$

The normalizing constant of the full conditional (3.3) does not have a closed form and hence it is not possible to sample directly from this distribution. Therefore, to generate from this full conditional probability density it is used a Metropolis-Hastings step. The idea of the Metropolis-Hastings algorithm is to use an auxiliary and simpler distribution, called proposal distribution, from which we know how to sample directly. The proposed value typically depends on the current configuration making the successive random draws stochastically dependent.

In [19], the proposal distribution for the a word location is a bivariate Gaussian density

centered at the current word position. However, other proposal distributions can be used in order to speed up the algorithm convergence. In the next subsection, we present an alternative proposal distributions to the bivariate Gaussian. We discuss the advantages and disadvantages of the new proposal generators and compare them.

3.1 Proposal Distribution

In this section, we present one distribution that can be used as an alternative to Gaussian distribution in COWORDS algorithm. Unlike Gaussian distribution, this new density is discrete and asymmetric. The rationale behind this proposal is to take into account not only the word position but also the empty spaces in the word cloud and their distances from the word cloud center. This new distribution puts higher probability mass in empty regions that are close to word cloud center, while regions that are far from the center and more completely filled will have lower or zero density.

Let L be a $[0, l]^2$ continuous region where the words will be located. We superimpose on L the rectangular grid G_z controlled by the positive integer z, the number of vertical and horizontal equispaced divisions. In Figure 3.3 we have $L = [0, 1]^2$ and $z = 21$. We denote by $C_{(i,j)}$ the cells in L determined by the grid G_z, for $i, j \in \{1, \cdots, z\}$. In order to find and select empty regions, we define a neighborhood around each cell and count its number of empty neighbors. The function to perform this task is $\mathrm{N}_J(i, j)$, where J is the window size around $C_{(i,j)}$ containing all neighbors of $C_{(i,j)}$. In Figure 3.3, we show two different window sizes for cell $C_{(18,6)}$, $J = 1$ and $J = 2$. The first one, with $J = 1$, is represented by a hatched area, while the neighborhood generated with $J = 2$ is illustrated through a dashed bounding box. For each cell that belongs to the neighborhood, we set a weight ϕ_{ij} depending on whether the cell is empty or not. In Figure 3.3, $\mathrm{N}_1(18, 6) = 8$ and $\mathrm{N}_2(18, 6) = 21$. The Euclidean distance $|C_{ij} - C_{cc}|$ between the central cell and the cell $C_{(18,6)}$ is shown as a dashed segment that links both cases. The words are represented by gray rectangles.

The main idea of our proposed distribution is to generate words position in the Metropolis-Hastings proposal algorithm with a higher probability of acceptance. As a consequence, this implies in a faster convergence of COWORDS algorithm. Instead of using a proposed distribution that takes into account only the current position of the word, we also consider the empty regions and the center of the cloud.

After selecting a grid cell C_{ij} to contain the k-th word, we drawn an uniform point inside the cell to be the word center. Let $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ be the positions of the words centers in the cloud. The Gibbs sampler step uses the conditional probability of observing the k-th word center at w_k given all the other words positions $\mathbf{W}_{-k} = (w_1, w_2, \ldots, w_{k-1}, w_{k+1}, \ldots, w_n)$. This conditional probability distribution is shown in Equation (3.4), conditioned also on the windows size J and the parameter λ.

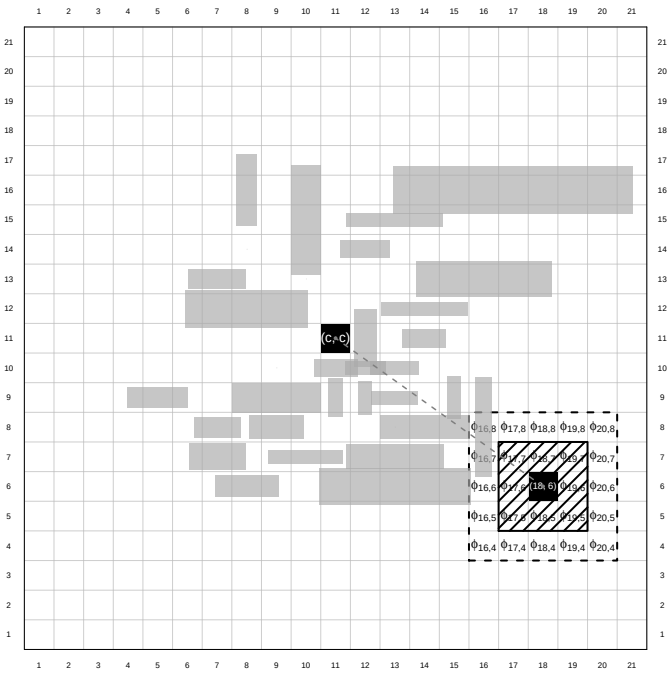


Figure 3.3: Example of the grid G_{21} to exemplify the proposal distribution.

$$\pi_1(w_k \mid \mathbf{W}_{-k}, J, \lambda) \propto \exp\left\{\log\left[\frac{\mathrm{N}_J(C_{ij})}{(2J+1)^2}\right] - \lambda|C_{ij} - C_{cc}|\right\} \text{ for } w_k \in C_{ij} \tag{3.4}$$

$$N_J(C_{ij}) = \sum_{(i,j)\in J} \phi_{ij} \tag{3.5}$$

where the weight ϕ_{ij}, in this case, is an indicator function:

$$\phi_{ij} = \begin{cases} 1, & \text{if the cell is empty.} \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

The λ parameter plays the role of controlling the distance weight between cell C_{ij} and the central cell C_{cc}.

To exemplify the different shapes of the probability distribution in Equation (3.4), we show in the Figure 3.4 the density for some values of λ and J. To build Figure 3.4 we removed the most frequent word from Figure 3.3 and then computed the probability of each cell. In this example, we considered a grid G_{100}. The Figure 3.3 is a matrix plot, where the rows are the window sizes and the columns are the λ parameters. The most bright regions indicate the cells most likely to be selected based on Equation 3.4. Notice that, for the results with $\lambda = 4$, the peripheral cells have a lower probability of being selected, whereas the central cells have a higher probability. For $\lambda = 4$, increasing value of J lead to higher probability of selecting cells already filled by rectangles, mainly those places in the boundary of the rectangles. To better understand this effect, consider a cell located in the center of a rectangle. If the window J is small, it is likely that all neighboring cells are filled and consequently $\pi_1 = 0$.

However, as J increases, it becomes more likely to reach empty cells thus increasing π_1.

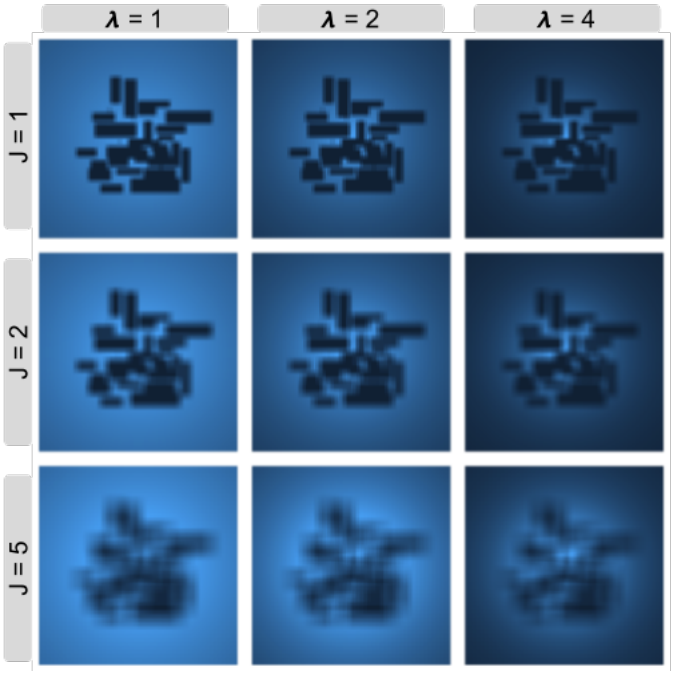


Figure 3.4: Proposal probability distribution for different parameters.

Given a value generated from the proposal distribution in Equation 3.4, an acceptance probability is calculated to accept or reject it as a value generated from the target distribution given by Equation (3.3). The algorithm works as follows: for each step m and word k, a value w^* is drawn from a proposal distribution $\mathbf{Q}$ conditioned at the current k-th word position $w_k^{(m-1)}$ and is computed the acceptance probability:

$$\rho\left(w_k^{(m-1)}, w^*\right) = \min\left\{1, \frac{\pi_c\left(w^*|\mathbf{W}_{-k}\right)\mathbf{Q}\left(w_k^{(m-1)}; w^*\right)}{\pi_c\left(w_k^{(m-1)}|\mathbf{W}_{-k}\right)\mathbf{Q}\left(w^*; w_k^{(m-1)}\right)}\right\}.$$

We then accept w^* as the new value $w_k^{(m)}$ with probability $\rho(w_k^{(m-1)}, w^*)$. If w^* is not accepted, we maintain $w_k^{(m)} = w_k^{(m-1)}$. When $\mathbf{Q}$ is equal to the Gaussian distribution as in the original COWORDS algorithm, the property of symmetry around the mean is used and we can simplify $\rho\left(w_k^{(m-1)}, w^*\right)$ by canceling out the $\mathbf{Q}$ density factor in both numerator and denominator as they are equal for this specific choice of proposal distribution. However, the probability distributions in Equation (3.4) does not hold this symmetry property, and therefore $\rho\left(w_k^{(m-1)}, w^*\right)$ cannot be simplified.

We used this new algorithm only for the set of words $\mathbf{W^s}$ that appear in more than one time or group. The set $\mathbf{W^u}$ of remaining words, those appearing in only one single word cloud, is placed in its respective word cloud by running the Wordle algorithm independently. This modification provides a faster algorithm and more tight word clouds. The complete algorithm is shown as Algorithm 2.

Algorithm 2 COWORDS algorithm

Require: Number of iterations M and optionally an initial configuration.
Ensure: Final position of words $\mathbf{W} = \{w_k\}_{k\in\{1,\dots,n\}}$.

$\mathbf{W} = \mathbf{W^s} \cup \mathbf{W^u}$
if starting position not set **then**
  for all $w_k \in \mathbf{W}^s$ **do**
    Generate $\mathbf{w}_k^{(0)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  end for
end if
for all $m \in \{1, \dots, M\}$ **do**
  for all $w_k \in \mathbf{W}^s$ **do**
    $w^* \sim \mathbf{Q}\left(w_k^{(m-1)}; w^*\right)$
    Take $\rho\left(w_k^{(m-1)}, w^*\right) = \min\left\{1, \frac{\pi_c\left(w^*|\mathbf{W}^s_{-k}\right)\mathbf{Q}\left(w_k^{(m-1)};w^*\right)}{\pi_c\left(w_k^{(m-1)}|\mathbf{W}^s_{-k}\right)\mathbf{Q}\left(w^*;w_k^{(m-1)}\right)}\right\}$.
    Generate $u \sim \mathcal{U}(0,1)$
    if $u < \rho\left(w_k^{(m-1)}, w^*\right)$ **then**
      $w_k^{(m)} \leftarrow w^*$
    else
      $w_k^{(m)} \leftarrow w_k^{(m-1)}$
    end if
  end for
end for

3.1.1 New Proposal Distribution Comparison

In order to compare our new proposal distribution π_1 against the Gaussian distribution, we use three Barack Obama's speeches. We generate simulated scenarios with different values of λ and J parameters and assess the solutions of both distributions based on two metrics.

The first metric is usually called *compactness* and is defined as

$$\delta = 1 - \frac{\text{used area}}{\text{total area}},$$

where "used area" represents the sum of the area occupied by all rectangles wrapping the words, and "total area" is the surface of the convex hull involving all the rectangles. Note that, when $\delta = 0$, the best possible packing has been obtained. The second metric corresponds to the summation of all squared distances measured between each pair of rectangle wrapping words for all possible times. This measure is given by $\boldsymbol{\alpha} = \sum_{t=1}^{T}\sum_{i\sim j}\alpha_{tij}^2$. Both metrics measure how much tightened the words are in the cloud.

In Figure 3.5, we show the value of δ for twelve tested scenarios. The COWORDS algorithm was run with $1,000$ iterations using three word clouds from three time periods, 50 words in each time and a grid G_{50} with 50 cells. In all sets, we used the last iteration to compute δ. Each column in Figure 3.5 displays a moment of the Obama's speech previously mentioned and the rows represent the variation of window J. The values of δ for the Gaussian distribution are replicated for each J to help the visual comparison of results. Qualitatively, the word clouds produced by the Gaussian distribution were visually similar to those produced by π_1. Quantitatively, the distributions also produced similar results, in the sense that for all cases compact word clouds were generated with no method being uniformly better than the other. The new proposal distribution with different parameters performed better in most scenarios.

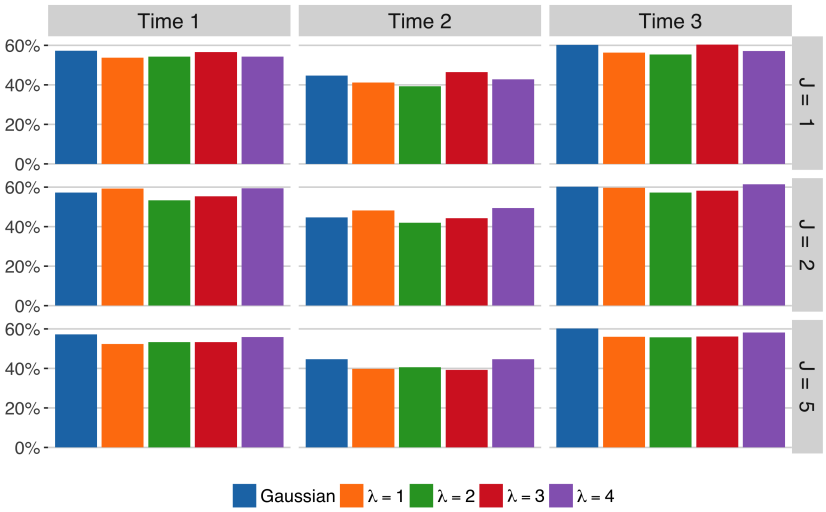
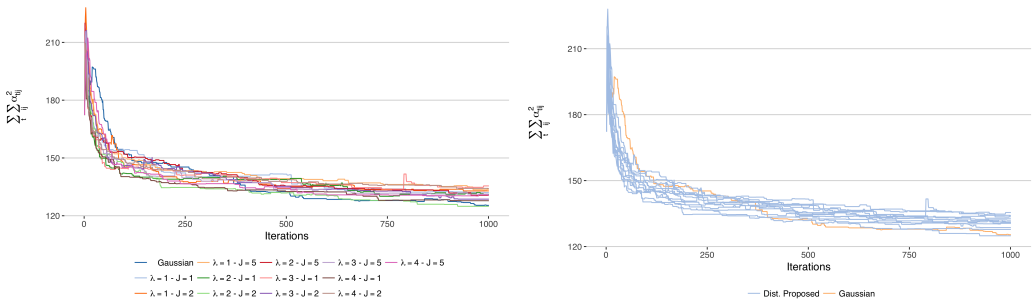


Figure 3.5: Comparison between different proposal distribution for COWORDS algorithm. The bar corresponds to the value of δ, that is the proportion of empty space inside the word cloud region.

In Figure 3.6 we show the traceplot for $\boldsymbol{\alpha}$ across $1,000$ iterations and 7 scenarios. Both Figures 3.6(a) and 3.6(b) present the same information, however we have used different colors to help the comparison between the methods. In this case, we always started the COWORDS algorithm with the same initial configuration. This initial configuration may have overlapping among the words. Therefore, the first iterations at each scenario is basically trying to remove all overlappings, increasing the value of $\boldsymbol{\alpha}$ in these iterations. Immediately after these iterations, the methods start to look for compact word clouds and thus the distances between the words decrease. When the COWORDS algorithm is set to our proposed distribution it tends to decrease $\boldsymbol{\alpha}$ faster than when using the Gaussian distribution. We can better visualize this behavior in Figure 3.6(b). Although such quick decrease occurs at the beginning, it does not keep decreasing so fast for all iterations. In the last iteration, the best result was obtained for π_1 with $\lambda = 1$ and $J = 2$. Nonetheless, the Gaussian distribution achieved the runner-up $\boldsymbol{\alpha}$ among the proposed configurations.



(a) Traceplot for $\boldsymbol{\alpha}$ to different parameters of π_i.

(b) Traceplot for $\boldsymbol{\alpha}$ comparing the Gaussian distribution and the π_1 distribution.

Figure 3.6: Simulation study with 50 words and $T = 3$ using the most frequent words of Barack Obama first speech, in Oct 13, 2012.

We can not conclude that our proposed distribution is uniformly better than Gaussian

distribution. However, we have evidence that π_1 may improve the convergence time. In this experiment, we highlight the main appeal of COWORDS which is the great flexibility provided by its probabilistic approach.

3.1.2 Case study: Game of Thrones

In order to visually illustrate the new improvements of the COWORDS algorithm we show an example generated by COWORDS. We collected the subtitles of the Game of Thrones series, which is one the most popular television series in the world. According to New York Times, in season seven, the average numbers of viewer was over 30 million per episode across all platforms. Figure 3.7 shows the 50 most frequent words from each seasons' subtitles as a result of running COWORDS for $1,000$ iterations which take two seconds. In the first three seasons the word "stark" appears in the 50 most frequent words, as well as, "lord", "king" and "father". The family Stark had a great importance in this first three seasons mostly because of Ned Stark, who is the hand of the king, his son Robb Stark, and his wife Catelyn Stark. Despite the popularity of the Stark family, in the fourth season the word "stark" does not appear among the most frequent words. Several factors in the series led to the low frequency. First, Ned Stark who was very important in the first season got killed at the first season end. In the next season, Robb Stark became the North king but, in the third season, he and his mother were killed. Then, in the two next seasons the Stark family does not have significant importance. However, in the sixth season, they regain importance with the Stark sisters Sansa and Arya. Another highlight in Figure 3.7 is the increase of the word "queen", which is not one of the most frequent words in the four first seasons, but increases in the seasons 5 and 6. In the last seasons, the female characters begin to hold high positions and take the central stage in the series. As a consequence, their importance in the series grow. For instance, Daenerys Targaryen is the Queen of Meereen and Cersei Lannister is the Queen Regent of the Seven Kingdoms.

4 COWORDS: Semantic Word Clouds

In the previous section, we presented COWORDS results for different proposal distributions, demonstrating that it is a flexible and easy-to-extend algorithm. In this section, we present another extension in which the positioning of words in the cloud must take a given context into account. The novelty in this work is to take both temporal and semantic contexts into account simultaneously. To motivate this analysis we collected approximately 2000 tweets for each of the years from 2010 to 2015 related to Dengue disease. These tweets have been manually labeled into 5 categories according to the content in its text: disease personal experience, campaign against the disease, information about the disease, personal

Accessed on November 2017: https://www.nytimes.com/2017/08/28/arts/television/game-of-thrones-finale-sets-ratings-record.html

Figure 3.7: Most frequent words from the subtitles of first six seasons of Game of Thrones.

opinion or joke. This taxonomy follows previous works using tweets to predict dengue incidence [22; 21]. In our case, we would like to visualize the evolution of the most tweeted words over time and their distribution in the respective classes.

4.1 COWORDS algorithm for semantic word clouds

The generalization of the COWORDS algorithm is constructed by adding a parameter β_{ij} as shown in Equation (3.7). The parameter β_{ij} is a positive parameter and can assume two values as defined in Equation (3.8). In this case, $\beta_{ij} = \kappa$ is the attraction force between correlated words and $\beta_{ij} = 2$ represents the attraction force between all other words. This parameter simultaneously guarantees compact word clouds and that correlated words will be positioned close to each other in the word cloud.

$$\pi(\mathbf{W}) \propto \exp\left\{-\sum_{t=1}^{T}\sum_{i\leq j}\alpha_{tij}^{\beta_{ij}}\right\} \times \prod_{t=1}^{T}\prod_{i\neq j}\mathbb{1}_{[S_{tij}]} \tag{3.7}$$

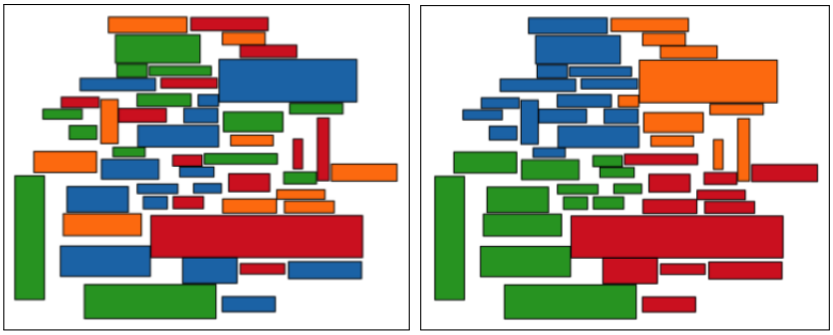
where β_{ij} is the indicator function

$$\beta_{ij} = \begin{cases} \kappa, & \text{if there is semantic correlation between the words } i \text{ and } j. \\ 2, & \text{otherwise.} \end{cases} \tag{3.8}$$

The probability density function in Equation (3.7) puts more probability mass in the word configurations that are more compact and has the same class words near each other. Figure 3.8 shows two possibles word configuration, where the color encode the classes. Figure

3.8(a) shows a configuration that has low probability of being observed from the probability density function in Equation (3.7). The configuration in Figure 3.8(b) has a much higher probability of being selected. In the COWORDS model, given by Equation (3.1), if we switch the positions of any two words of the same size, we get the same value of (3.1). However, in the model given in Equation (3.7), we only get the same value for (3.7) if the two exchanged words belong to the same class.

We draw sample configurations from (3.7) and the more satisfactory is a solution, is the higher the probability that the configuration is selected. The algorithm to sample from (3.7) is the Gibbs sampler with Metropolis step [17]. In order to evaluate this new extension of COWORDS, we performed a simulation study that is presented in the next section.



(a) A typical word cloud. (b) Semantic word cloud.

Figure 3.8: Two configurations that are possible according to the distribution given by Eq. (3.7). The cloud on the right-hand side has much higher probability density than the cloud on the left-hand side.

4.2 Evaluation of the semantic algorithm

In this section, we present different scenarios to evaluate the COWORDS ability in generating semantic word clouds. For this study, we create 36 scenarios varying according to some parameters. These parameters settings are: $T = \{1, 2, 4, 8\}$, $C = \{2, 4, 8\}$, $W = \{25, 50, 100\}$, where T is number of times, C the number of word classes and W the number of words in each word cloud. To simulate the content for each time, we randomly drawn words from Obama's speech. The words' class is randomly selected from the set of classes with equal probability. W is the cardinality of the frequent words set.

To assess the semantic word clouds generated by COWORDS we propose a metric. For each word i, we select its k_i nearest neighbors, where k_i is the total number of words belonging to the same class as i. For instance, if the class of i has 20 words, we select the 20 nearest words of i, irrespective of their classes. Next, we compute the proportion of these k_i nearest neighbors that belong to the class of i. When this proportion increases, we have a good configuration. When $T > 1$, we compute the average of this metric over all times. In all scenarios, the value of κ that performed better was 10.

In Table 3.1, we show the results for the proposed metric related to the 36 scenarios. When the number of classes increases and the number of words decreases, the performance

		Words		
Class	Time	25	50	100
2	1	96.0%	95.3%	85.8%
	2	91.3%	87.3%	96.0%
	4	92.8%	79.2%	85.8%
	8	87.2%	92.1%	76.2%
4	1	66.0%	72.3%	83.0%
	2	70.7%	78.8%	77.0%
	4	72.5%	76.6%	77.7%
	8	73.2%	80.6%	70.1%
8	1	44.0%	55.0%	62.7%
	2	51.3%	58.2%	63.2%
	4	52.3%	57.1%	65.3%
	8	36.7%	68.5%	67.3%

Table 3.1: Realized adjacencies metric to evaluate the COWORDS algorithm to generate semantic word cloud.

of the algorithm also decreases. We anticipate this effect since the number of words in each class will be small. However, as the number of words in the cloud increases the algorithm performed better. In order to visually exemplify the algorithm, we present an example in the next section.

4.3 Case Study

Dengue is an infectious disease that occurs in tropical countries such as Brazil, and is one of the major concerns for Brazilian public health officials. In this section, we visualize approximately $14,000$ tweets collected from 2010 to 2015 that contain words related to Dengue. Figure 3.9 shows a word cloud generated by COWORDS for the tweets over the five years. In this word cloud the colors of the words are randomly chosen, and for each time the 50 most frequent words were selected. Clearly, some important words begin to emerge. For instance, the word "vacina" ("vaccine") rises in 2014 and keeps increasing its frequency in 2015. Another relevant words are "chikungunya" and "zika", that are others disease transmitted by the same mosquito, the *Aedes Aegypti*. Zika and Chikungunya are highlighted in the years 2014 and 2015, when a huge outbreak started in Brazil.

As previously mentioned, these tweets are manually labeled into five classes: *campaign* that are tweets usually tweeted by Brazilian public health officials to prevent the disease; *personal experience* are tweets that the user expresses some personal experience with the disease; *information* express information about epidemic or number of people with the disease; *opinion* are tweets from users that give their opinion about the disease; *joke* are jokes related to dengue. Figure 3.10 shows the word clouds with the 50 most frequent words for each class of tweets. The words frequency of the class *campaign* and *information* are similar, however, are quite different from the other class. In the *campaign* and *information* the words

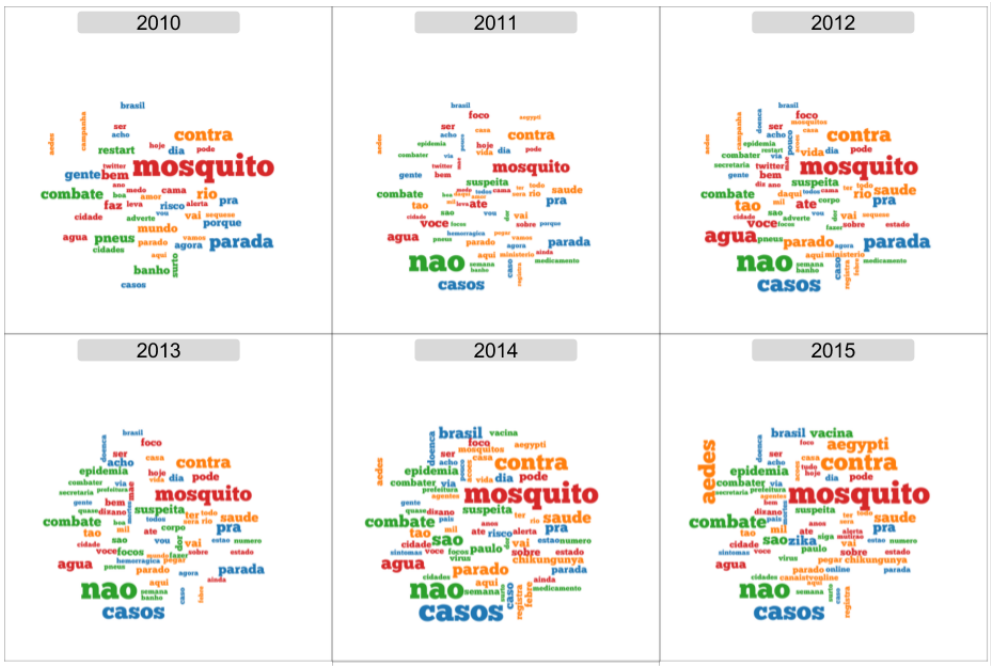


Figure 3.9: Word clouds generated by COWORDS algorithm for tweets related to Dengue disease that are collected between 2010 and 2015.

"combate" and "contra" that express fighting the disease. While the word "casos" means "dengue cases" in the class *information* have a great frequency. In the class *experience*, the word "suspeita" that means "suspicion" appears in the most frequency words, which mean that the user has a suspicion of dengue infection.

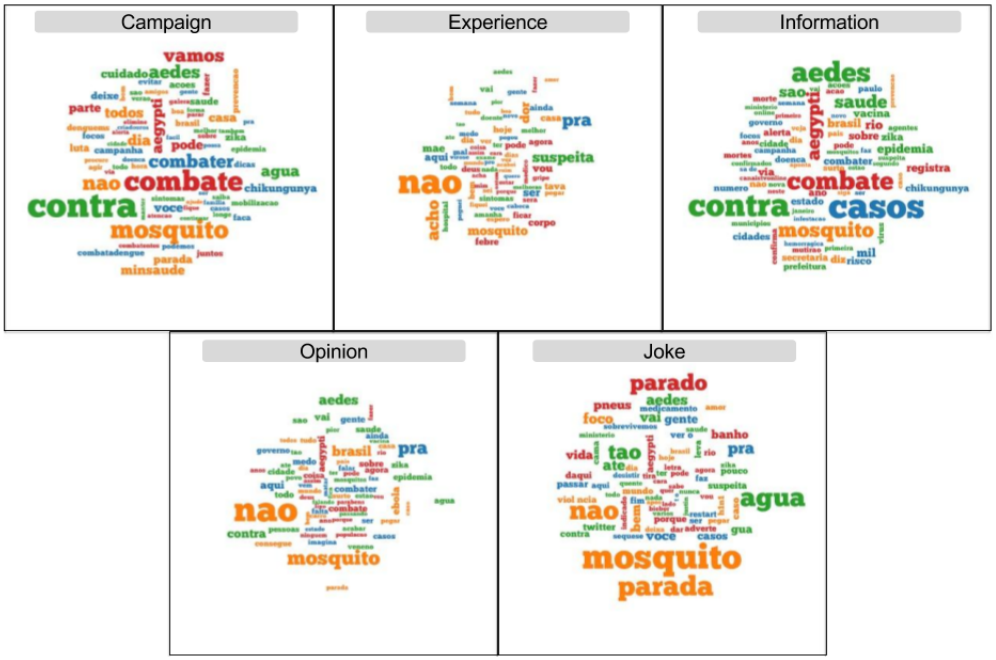


Figure 3.10: Most frequent words of the tweets collected from 2010 to 2015 sorted in five classes.

Ideally, we want to jointly visualize the two dimensions: temporal and contextual. Therefore, we apply the COWORDS algorithm to generate semantic word clouds considering the classes of tweets as our semantic dimension. Words from tweets that belong to the same

class are considered semantically correlated. The output from COWORDS to semantic word cloud is presented in Figure 3.11. To generate this visualization, we considered the 20 most frequent words for each class in each year. In addition, we present tweets collected in 2016. The result is a kind of merge between the two word clouds shown in Figures 3.9 and 3.10.

5 Conclusions and Future Work

In this paper, we demonstrated the flexibility of the COWORDS to add new constraints in its formulation. We proposed an extension of COWORDS for generating semantic word cloud across time that preserves the COWORDS principles. That is, the clouds are easily comparable because the shared words are placed in the same location in all clouds. In addition, the words that are semantically correlated are placed close to each other in all word clouds. Another contribution is a new proposal distribution as an alternative to the Gaussian distribution. This new distribution improved the convergence and performance of COWORDS. The running strategy of COWORDS was modified to run only for shared words and, for other words, we run the Wordle algorithm. These modifications improved and augmented the COWORDS application range. We demonstrated the effectiveness of our method through simulation and the presentation of case studies. As future work, the shape of the word clouds can be previously defined and incorporated in the constraints. As consequence, the word clouds can be generated across time with a specific shape.

References

[1] L. Barth, S.G. Kobourov, and S. Pupyrev, *Experimental comparison of semantic word clouds*, in *Experimental Algorithms*, Springer, 2014, pp. 247–258.

[2] S. Bateman, C. Gutwin, and M. Nacenta, *Seeing things in the clouds: the effect of visual features on tag cloud selections*, in *Proceedings of the nineteenth ACM conference on Hypertext and hypermedia*. ACM, 2008, pp. 193–202.

[3] M. Burch, S. Lohmann, F. Beck, N. Rodriguez, L. Di Silvestro, and D. Weiskopf, *RadCloud: Visualizing multiple texts with merged word clouds*, in *Information Visualisation (IV), 2014 18th International Conference on*. IEEE, 2014, pp. 108–113.

[4] Q. Castellà and C. Sutton, *Word Storms: Multiples of Word Clouds for Visual Comparison of Documents*, in *Proceedings of the 23rd International Conference on World Wide Web*, Republic and Canton of Geneva, Switzerland. International World Wide Web Conferences Steering Committee, WWW '14, 2014, pp. 665–676. Available at http://dx.doi.org/10.1145/2566486.2567977.

[5] M.T. Chi, S.S. Lin, S.Y. Chen, C.H. Lin, and T.Y. Lee, *Morphable word clouds for time-varying text data visualization*, IEEE transactions on visualization and computer graphics 21 (2015), pp. 1415–1426.

[6] C. Collins, F. Viegas, and M. Wattenberg, *Parallel Tag Clouds to explore and analyze faceted text corpora*, in *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, Oct. 2009, pp. 91–98.

[7] W. Cui, Y. Wu, S. Liu, F. Wei, M.X. Zhou, and H. Qu, *Context preserving dynamic word cloud visualization*, in *Pacific Visualization Symposium (PacificVis), 2010 IEEE*. IEEE, 2010, pp. 121–128.

[8] S. Deutsch, J. Schrammel, and M. Tscheligi, *Comparing Different Layouts of Tag Clouds: Findings on Visual Perception.*, in *HCIV*. Springer, 2009, pp. 23–37.

[9] M.J. Halvey and M.T. Keane, *An assessment of tag presentation techniques*, in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1313–1314.

[10] M.A. Hearst, *What's up with tag clouds?*, Visual Business Intelligence Newsletter (2008), pp. 1–5.

[11] K. Koh, B. Lee, B. Kim, and J. Seo, *Maniwordle: Providing flexible control over wordle*, IEEE Transactions on Visualization and Computer Graphics 16 (2010), pp. 1190–1197.

[12] B.Y. Kuo, T. Hentrich, B.M. Good, and M.D. Wilkinson, *Tag clouds for summarizing web search results*, in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 1203–1204.

[13] B. Lee, N.H. Riche, A.K. Karlson, and S. Carpendale, *Sparkclouds: Visualizing trends in tag clouds*, Visualization and Computer Graphics, IEEE Transactions on 16 (2010), pp. 1182–1189.

[14] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, and J. Dean, *Distributed representations of words and phrases and their compositionality*, in *Advances in neural information processing systems*. 2013, pp. 3111–3119.

[15] F.V. Paulovich, F. Toledo, G.P. Telles, R. Minghim, and L.G. Nonato, *Semantic wordification of document collections*, in *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 2012, pp. 1145–1153.

[16] A.W. Rivadeneira, D.M. Gruen, M.J. Muller, and D.R. Millen, *Getting our head in the clouds: toward evaluation studies of tagclouds*, in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2007, pp. 995–998.

[17] C.P. Robert and G. Casella, *Monte Carlo Statistical Methods (Springer Texts in Statistics)*, Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[18] J. Schrammel, M. Leitner, and M. Tscheligi, *Semantically structured tag clouds: an empirical evaluation of clustered presentation approaches*, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 2037–2040.

[19] L.G.S. Silva and R.M.A. ao, *Cowords: A probabilistic model for multiple word clouds*, Journal of Applied Statistics ("in press").

[20] J. Sinclair and M. Cardew-Hall, *The folksonomy tag cloud: when is it useful?*, Journal of Information Science 34 (2008), pp. 15–29.

[21] R.C.S.N.P. Souza, R.M. Assunção, D.M. de Oliveira, D.E.F. de Brito, and W. Meira, *Infection Hot Spot Mining from Social Media Trajectories*, in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II*, P. Frasconi, N. Landwehr, G. Manco, and J. Vreeken, eds., Springer International Publishing, Cham (2016), pp. 739–755.

[22] R.C.S.N.P. Souza, D.E.F. de Brito, R.L. Cardoso, D.M. de Oliveira, W. Meira, and G.L. Pappa, *An Evolutionary Methodology for Handling Data Scarcity and Noise in Monitoring Real Events from Social Media Data*, in *Advances in Artificial Intelligence – IBERAMIA 2014: 14th Ibero-American Conference on AI, Santiago de Chile, Chile, November 24-27, 2014, Proceedings*, A.L. Bazzan and K. Pichara, eds., Springer International Publishing, Cham (2014), pp. 295–306.

[23] J. Steele and N. Iliinsky, *Beautiful Visualization: Looking at Data Through the Eyes of Experts*, 1st ed., O'Reilly Media, Inc., 2010.

[24] H. Strobelt, D. Oelke, C. Rohrdantz, A. Stoffel, D.A. Keim, and O. Deussen, *Document cards: A top trumps visualization for documents*, IEEE Transactions on Visualization and Computer Graphics 15 (2009), pp. 1145–1152.

[25] E.R. Tufte, *Beautiful evidence, vol. 23* (2006).

[26] F. Viegas, M. Wattenberg, and J. Feinberg, *Participatory visualization with wordle*, Visualization and Computer Graphics, IEEE Transactions on 15 (2009), pp. 1137–1144.

[27] Y. Wang, X. Chu, C. Bao, L. Zhu, O. Deussen, B. Chen, and M. Sedlmair, *Edwordle: Consistency-preserving word cloud editing*, IEEE Transactions on Visualization and Computer Graphics (2017).

[28] Y. Wu, T. Provan, F. Wei, S. Liu, and K.L. Ma, *Semantic-preserving word clouds by seam carving*, in *Computer Graphics Forum*, Vol. 30. Wiley Online Library, 2011, pp. 741–750.

[29] J. Xu, Y. Tao, and H. Lin, *Semantic word cloud generation based on word embeddings*, in *Pacific Visualization Symposium (PacificVis), 2016 IEEE*. IEEE, 2016, pp. 239–243.

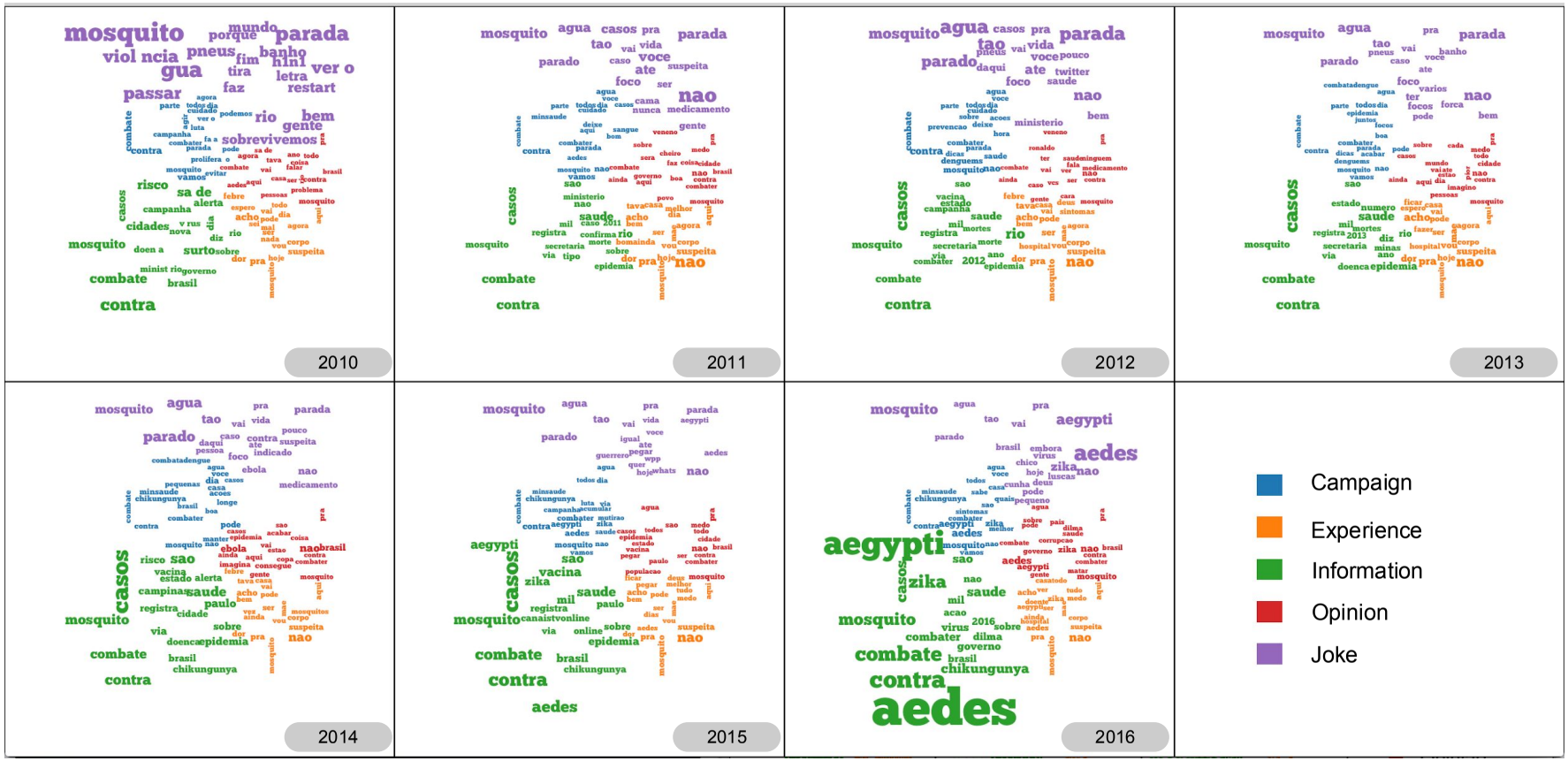


Figure 3.11: The output from the COWORDS algorithm modified to generate semantic word cloud. The word clouds show the most frequent word in each class of the words over the time.