Instituto Politécnico de Coimbra
Instituto Superior de Contabilidade
e Administração de Coimbra

Alcides de Almeida Seiça

Aplicação de técnicas de *Text Mining* na perceção dos cidadãos quanto ao funcionamento da Autoridade Tributária e Aduaneira

Coimbra, Outubro de 2019

Aplicação de Técnicas de *Text Mining* na perceção dos cidadãos quanto ao funcionamento da Autoridade Tributária e Aduaneira
Alcides de Almeida Seiça
ISCAC | 2019

Instituto Politécnico de Coimbra
Instituto Superior de Contabilidade
e Administração de Coimbra

Alcides de Almeida Seiça

Aplicação de técnicas de *Text Mining* na perceção dos cidadãos quanto ao funcionamento da Autoridade Tributária e Aduaneira

Dissertação submetida ao Instituto Superior de Contabilidade e Administração de Coimbra para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Análise de Dados e Sistemas de Apoio à Decisão, realizado sob a orientação do Professor Doutor António Rui Trigo Ribeiro e coorientação do Professor Doutor Fernando Paulo dos Santos Rodrigues Belfo.

Coimbra, Outubro de 2019

TERMO DE RESPONSABILIDADE

Declaro ser o autor desta dissertação, que constitui um trabalho original e inédito, que nunca foi submetido a outra Instituição de ensino superior para obtenção de um grau académico ou outra habilitação. Atesto ainda que todas as citações estão devidamente identificadas e que tenho consciência de que o plágio constitui uma grave falta de ética, que poderá resultar na anulação da presente dissertação.

DEDICATÓRIA

À minha família.

AGRADECIMENTOS

Os agradecimentos iniciais vão para todos os docentes e colegas da primeira edição deste mestrado, por juntos termos desbravado os caminhos para a sua reedição certamente com outros avanços e novos conhecimentos.

Expresso também os meus agradecimentos pelo apoio manifestado pelos colegas de trabalho com quem partilhei as muitas dificuldades e pequenos sucessos.

Termino com um especial agradecimento aos meus orientadores Professor Doutor António Trigo e Professor Doutor Fernando Belfo pelos valiosos contributos, comentários e incentivos.

RESUMO

A crescente importância das redes sociais na nossa sociedade leva governos e instituições públicas a privilegiarem estas redes, não só, na comunicação com os seus cidadãos, mas também na perceção da opinião e grau de satisfação que os cidadãos têm sobre os serviços prestados.

Recorrendo a técnicas de *Text Mining*, mais especificamente, de Análise de Sentimentos (*Sentiment Analysis* ou *Opinion Mining*), é possível extrair informação útil das redes sociais que permita a identificação e acompanhamento das opiniões dos cidadãos.

O objetivo principal deste trabalho consiste na aplicação de técnicas de Análise de Sentimentos a um conjunto de *posts* publicados na rede social Twitter que expressem opiniões acerca do funcionamento geral das autoridades fiscais portuguesas, tendo em vista a apresentação de novos contributos para a melhoria da interação daqueles Serviços com os contribuintes. A fase de extração dos *tweets* correspondeu a um período de sete meses, entre agosto de 2018 e fevereiro de 2019.

Como principais resultados destacamos a criação de uma nova abordagem bietápica para a Análise de Sentimentos designada de LexiNB e a criação de um algoritmo que permite automatizar o tratamento das opiniões dos utilizadores do Twitter, visando a evolução da organização para um estádio de comunicação bidirecional nas redes sociais.

Palavras-chave: Autoridade tributária; *Text Mining*; Análise de Sentimentos; LexiNB; Twitter

ABSTRACT

The growing importance of social networks in our society leads governments and public institutions to privilege these networks, not only in communicating with their citizens, but also in the perception of citizens' opinion and degree of satisfaction with the services provided.

By using Text Mining techniques, more specifically, of Sentiment Analysis or Opinion Mining, it is possible to extract useful information from social networks that allows the identification and monitoring of citizens' opinions.

The main objective of this work is the application of Sentiment Analysis techniques to a set of posts published in the social network Twitter that express opinions about the general functioning of the Portuguese tax authorities, in order to present new contributions to improve the interaction of those Services with taxpayers. The tweets extraction phase corresponded to a period of seven months, from August 2018 to February 2019.

As main results we highlight the creation of a new two-step approach to Sentiment Analysis called LexiNB and the creation of an algorithm that allows to automate the treatment of the opinions of Twitter users, aiming the evolution of the organization to a stage of two-way communication in the social networks.

Keywords: Tax Authorities; *Text Mining*; Sentiment Analysis; LexiNB; Twitter

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

Lista de abreviaturas, acrónimos e siglas

API – Application Programming Interface

CRAN – Comprehensive R Archive Network

CRISP-DM – Cross Industry Standard Process for Data Mining

CSV – Comma-Separated Values

DCBD – Descoberta de Conhecimento em Bases de Dados

IMDb – Internet Movie Database

IRC – Imposto sobre o Rendimento das Pessoas Coletivas

IRS – Imposto sobre o Rendimento das Pessoas Singulares

IVA – Imposto sobre o Valor Acrescentado

OCDE – Organização para a Cooperação e Desenvolvimento Económico

PCA – Principal Component Analysis

PLN – Processamento de Linguagem Natural

POS-Tagging – Part-of-Speech Tagging

SMTs – Social Media Technologies

SVD – Singular Value Decomposition

SVM – Support Vector Machine

TF-IDF – Term Frequency – Inverse Document Frequency

INTRODUÇÃO

O rápido crescimento e massificação da utilização da Internet e a enorme popularidade das redes sociais alteraram as formas de interação entre as pessoas e as organizações. A crescente quantidade de conteúdos publicados nas redes sociais sob a forma de mensagens ("*posts*") em que as pessoas expressam a sua opinião sobre os mais variados tipos e assuntos apresenta-se como um novo desafio para as organizações. As mensagens publicadas nas redes sociais podem manifestar satisfação ou duras críticas, refletindo os "estados de alma" do público em geral.

A rede social Twitter surgiu em 2006 e apresentava uma funcionalidade inovadora: cada *post*, geralmente designado por *tweet*, não deve exceder o limite de 140 caracteres. Devido à sua crescente popularidade, a rede passou por um processo de aumento gradual de eficiência e acessibilidade, tornando-se de grande importância, dando aos seus utilizadores o poder de criar e compartilhar ideias e informações instantaneamente, sem barreiras (TWITTER, 2017)[1]. A cada minuto, são publicados cerca de 360 mil *tweets* (TWITTER STATS, 2018) [2] e os mesmos dispõem de informações valiosas que possibilitam às organizações descobrir a opinião desses utilizadores em relação aos seus produtos e serviços (Filho, 2014).

Se atendermos ao elevado número de utilizadores das redes sociais, então o acompanhamento e monitorização deste processo apresenta-se como uma vantagem para as organizações, pois possibilita conhecer, em tempo útil, os problemas que possam vir a afetar a sua imagem. Ao analisar essas informações, percebeu-se que as organizações poderiam ter a vantagem de conhecer as opiniões dos utilizadores dos seus serviços ou produtos fornecidos a partir de dados das redes sociais (Gomes, 2012).

No sector público, uma área onde o desperdício de dinheiro tem um grande impacto em toda a sociedade e onde os indivíduos têm uma grande desconfiança quanto a verbas e a negócios pouco transparentes com outras organizações, a utilização e a análise destes dados pode trazer mais transparência, aumentar a produtividade e descobrir-se novas necessidades. De acordo com Kim, Trimi & Chung (2014), a implementação de

[1] https://twitter.com/twittersafety
[2] https://www.businessofapps.com/data/twitter-statistics/

aplicações de *Big Data* no setor público possibilita a ocorrência de melhorias no processo de decisão.

Os governos utilizam as redes sociais, devido à sua enorme popularidade, como forma de facilitação e iteratividade da sua comunicação, de alcançar novas audiências e colocar um rosto humano no governo. Espera-se que este processo permita reduzir a duplicação e economizar tempo e dinheiro. São diversas as utilizações que os governos procuram nas redes sociais nomeadamente para divulgação de:

- informações de emergência;
- informações de segurança e condições climatéricas;
- lembretes (pagamento de impostos, prazos de candidatura e outros);
- administração local – incêndios, alertas, construção de estradas, etc.

Na nota informativa do Forum on Tax Administration (2011), subordinado ao tema "*Social Media Technologies* (SMTs)", reconhece-se que o desenvolvimento e a utilização de SMTs se encontra ainda numa fase muito precoce e é útil refletir sobre a sua evolução.

A figura seguinte mostra um modelo de maturidade para a implementação e utilização das redes sociais como ferramentas de comunicação em organizações, como as administrações tributárias.

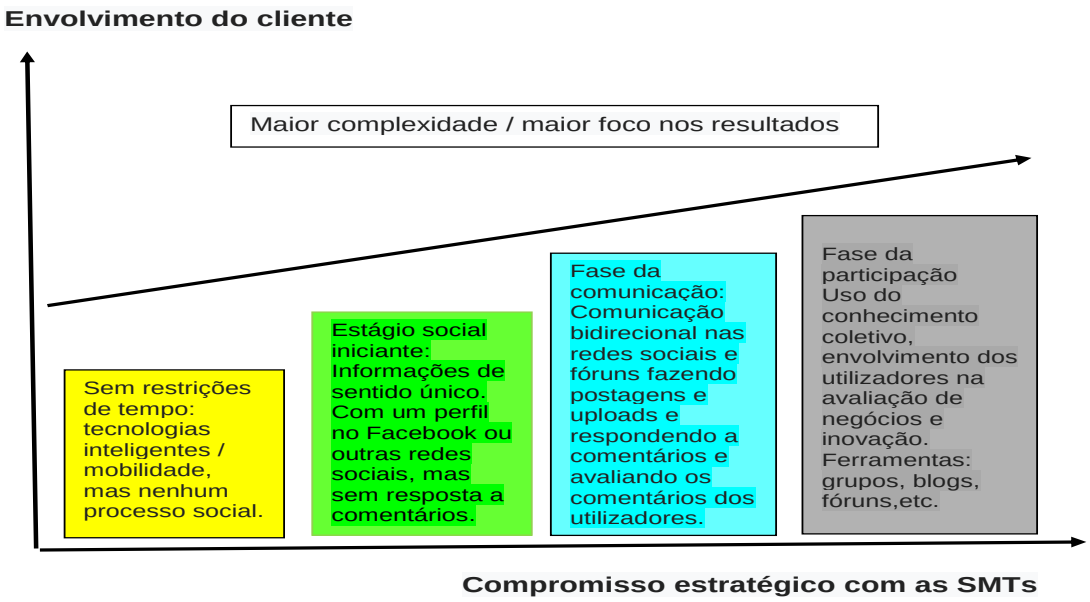


Figura 0.1 Modelo de maturidade na utilização de redes sociais

Fonte: Adaptado de Forum on Tax Administration (2011)

O modelo proposto retrata a evolução das redes sociais envolvendo o cliente/cidadão num grau cada vez mais complexo de interatividade, o que exige um maior envolvimento estratégico da administração fiscal. Num primeiro estádio, as redes sociais são utilizadas unicamente para transmissão de informação enquanto no estádio final se propõe a utilização do conhecimento partilhado e maior envolvimento na avaliação da organização. No entanto, a maior dificuldade consiste na criação de mecanismos que possibilitem a sistematização dessa informação por forma a torná-la útil.

As técnicas de aprendizagem-máquina assumem um papel de relevo, pois permitem identificar padrões e formas de utilização da informação a explorar. Com recurso às técnicas de Análise de Sentimentos, é possível acompanhar a evolução de um conceito ou opinião sobre determinado produto/serviço ao longo do tempo, permitindo avaliar o grau de satisfação dos clientes/cidadãos e avaliar os resultados das ações destinadas a influenciar essa opinião.

Este trabalho tem como objetivo geral, a aplicação de técnicas de Análise de Sentimentos a um conjunto de *posts* publicados na rede social Twitter que expressam opiniões relacionadas com o funcionamento geral das autoridades fiscais portuguesas (Administração Tributária e Aduaneira) e categorizar esses *tweets* de acordo com o sentimento expresso, utilizando técnicas de mapeamento das características de uma opinião no contexto escolhido, recorrendo às bibliotecas disponíveis para a linguagem "R".

Para atingir o objetivo proposto, elegem-se dois modelos de aprendizagem de máquina supervisionada (*supervised machine learning*), respetivamente *Naive Bayes* e *Support Vector Machines* (SVM), e um método baseado em léxicos em que será utilizada a biblioteca "LexiconPT" do software R. Propomo-nos também, efetuar uma análise comparativa dos resultados obtidos pelos diferentes modelos utilizados, na busca do que melhor se ajusta aos objetivos pretendidos.

Podem, então, formular-se as seguintes hipóteses iniciais para este estudo:

- Hipótese 1 – É possível classificar automaticamente os *tweets* relacionados com o funcionamento geral das autoridades fiscais portuguesas, de acordo com o sentimento expresso nas respetivas mensagens (positivas, negativas ou neutras) utilizando, para o efeito, técnicas de *machine learning*?

- Hipótese 2 – É possível fazer o acompanhamento e monitorização dessas manifestações de opinião, nomeadamente através da variação na tendência da classificação dos *tweets* de acordo com o sentimento expresso nas respetivas mensagens (positivas, negativas ou neutras) quanto ao momento da criação da mensagem ou em função de algum(ns) termo(s) utilizados na pesquisa?

Para melhor sistematização deste trabalho, o seu conteúdo foi dividido em três capítulos. Após a introdução, no capítulo 1 apresentamos uma revisão da literatura e dos trabalhos relacionados com pertinência para o estudo do caso. O capítulo 2 apresenta os aspetos genéricos da metodologia de investigação utilizada, nomeadamente no que diz respeito à recolha e pré-tratamento dos *tweets*. Reservámos o capítulo 3 para a descrição dos procedimentos de pré-processamento e apresentação dos resultados da análise sintática aplicada à globalidade dos *tweets* resultantes do pré-tratamento, bem como para a descrição das tarefas realizadas para a criação da amostra e de pré-processamento utilizados na geração dos modelos testados, finalizando com a discussão dos respetivos resultados. Terminamos o nosso trabalho com a descrição das principais conclusões obtidas e com a apresentação de perspetivas de desenvolvimento do trabalho que iniciámos com este projeto.

1 REVISÃO DA LITERATURA

A rápida expansão da internet provocou um aumento exponencial do volume de informação disponível sob a forma de opiniões discutidas em fóruns, comunidades e redes sociais. Indurkhya & Damerau (2010) citam que as opiniões são tão importantes que, onde quer que se queira tomar decisões, as pessoas querem ouvir a opinião de outros. Isto aplica-se não apenas às pessoas, mas também às organizações porquanto importa, a estas últimas, conhecer a opinião dos clientes/cidadãos acerca dos seus produtos e serviços.

A utilização das redes sociais por um elevado número de pessoas leva a que as organizações, administradores e governos reflitam cuidadosamente sobre como podem beneficiar também da sua utilização, usando essas ferramentas para ganhar maior confiança, dinamismo na divulgação de conteúdos e compreensão das preocupações dos seus clientes/cidadãos. Tal pensamento estratégico é importante para adotar a nova forma de interação com os indivíduos e para obter mais *insights* sobre as perceções e opiniões dos cidadãos em algumas questões (Khasawneh & Abu-Shanab, 2013). A este respeito, a nota informativa do "*Forum on Tax Administration*" (2011), refere que as SMTs são a face nova e personalizada da Internet. A sua chegada e desenvolvimento trazem promessas de contacto pessoal estratificado e novas formas de comunicação e interação com um número potencialmente grande e crescente de intervenientes do sistema fiscal.

1.1 Descoberta de Conhecimento em Bases de Dados

A Descoberta de Conhecimento em Bases de Dados (DCBD), foca-se na exploração computadorizada em grandes massas de dados para descobrir padrões interessantes entre elas (Feldman, et al., 1998). A maioria dos trabalhos de DCBD é suportada em bases de dados estruturadas, o que não é o caso da generalidade dos dados retirados de documentos de texto em que há necessidade de recorrer a técnicas auxiliares que possibilitem a sua estruturação. A extração de informação contida em dados não estruturados tem como objetivo identificar e extrair informações de eventos ou relacionamentos em textos de linguagem natural, construindo ainda uma representação que organiza essas informações obtidas (Grishman, 1998).

No mesmo sentido, Kushmerick & Thomas (2003) consideram a extração de dados retirados de documentos de texto como "uma forma superficial de processamento de documentos que envolvem o preenchimento de uma base de dados com valores

automaticamente extraídos de documentos". Esta definição enfatiza a extração de informação como um recurso utilizado na preparação dos dados a utilizar em tarefas posteriores, como por exemplo, usar como informação pré-processada para Análise de Sentimentos. Para chegar a esse resultado, as informações extraídas são geralmente organizadas num modelo, isto é, em estruturas compostas por campos, a preencher com os dados extraídos no processo de extração de informação (Álvarez, 2007). A extração da informação pode ser definida ainda como o processo de preencher uma base de dados estruturada a partir de fontes não estruturadas ou texto puro (Gaizauskas & Wilks, 1998).

As coleções de textos revestem a forma de dados não estruturados e provêm de múltiplas fontes com relacionamentos complexos e em permanente evolução. As bases de dados não estruturados rapidamente atingem uma enorme dimensão. Existe atualmente uma regra empírica apontando para o facto de os dados estruturados representarem apenas 20% das informações disponíveis para uma organização e que 80% de todos os dados estão em forma não estruturada[3]. O aproveitamento da informação contida nestes últimos representa o próximo desafio do *Big Data*.

Big Data é a designação usada para uma coleção de conjuntos de dados de uma dimensão tal que se torna difícil o seu processamento usando ferramentas de bases de dados manipuladas à mão ou aplicações tradicionais de processamento de dados. O processo inclui: recolher, tratar, armazenar, pesquisar, partilhar, transferir, analisar e visualizar. As soluções de *Big Data* recorrem a grandes volumes de dados procurando ajudar na deteção de padrões complexos, que seriam difíceis de observar sem a análise de grandes volumes de dados.

1.2 Processamento de Linguagem natural (PLN)

O Processamento de Linguagem Natural (PLN) é a abordagem informatizada que se baseia num conjunto de teorias e de tecnologias direcionadas para a análise e representação de textos que ocorrem naturalmente, num ou mais níveis de análise linguística, com a finalidade de obter processamento de linguagem para uma variedade de tarefas ou aplicações (Liddy, 2001), tradução livre.

[3] Ver https://pt.wikibooks.org/wiki/SQL/Dados_Estruturados,_Semi-Estruturados_e_Não_Estruturados

Figura 1.1 Fases da análise de PLN

Fonte: Adaptado de Indurkhya & Damerau (2010)

O processo de PLN representado na Figura 1.1 inicia-se com a segmentação por tokenização[4] das frases. A maior dificuldade do processo reside no facto dos textos em linguagem natural raramente serem compostos por frases curtas, bem formadas e delimitadas.

A análise lexical é a fase seguinte e consiste na análise do significado das palavras e de partes do diálogo recorrendo a combinações entre si, e mantendo as regras da sua formação tendo em vista o aumento da eficiência dos algoritmos e diminuição do espaço de armazenamento.

Segue-se a análise sintática que se debruça sobre os aspetos gramaticais e de estrutura das frases tendo em vista a extração do seu significado (Indurkhya & Damerau, 2010).

Na análise semântica, procura definir-se o significado de uma frase através de um objeto estruturado, passível de manipulação e interpretação subsequente. A semântica e o

[4] Tokenization é o ato de dividir uma sequência de carateres em pedaços, tais como palavras, frases, símbolos e outros elementos chamados *tokens*. No processo de tokenização, alguns caracteres como sinais de pontuação são descartados.

processamento ao nível do discurso têm sido menos estudados do que as questões sintáticas e não são facilmente aplicadas de forma generalizada.

A análise pragmática é o estádio final do processo de PLN e envolve a análise da linguagem considerando a influência do contexto comunicacional, extrapolando assim a visão da semântica e da sintaxe. É nesta fase que examinamos os resultados gerados pelo programa de computador, partindo da intenção inicial de comunicar algo para um estádio em que se determina o conteúdo do que foi dito, selecionando a redação e organização retórica e ajustando-a a uma gramática. Segundo Indurkhya & Damerau (2010), as técnicas de PLN mais utilizadas são:

- Segmentação de texto: palavras (*tokens*) ou frases;
- *Part-of-Speech Tagging* – aposição de etiquetas morfossintáticas nos tokens;
- Identificação de Entidades Nomeadas – identificação de nomes próprios, tais como nomes de pessoas, locais e organizações;
- Remoção de palavras não discriminantes (*stop words*);
- Normalização de palavras (acentuação, maiúsculas, *stemming* e *lemmatization*);
- Seleção de características – técnica para a redução de dimensionalidade que consiste na seleção de um conjunto de dimensões que melhor descrevem o problema, excluindo as restantes.

Para Loper & Bird (2002) o objetivo da etiquetagem morfossintática (*POS-Tagging*) é classificar os *tokens* de uma frase (ou texto) de acordo com suas classes morfológicas (substantivo, verbo, adjetivo, etc.).

POS-Tagging é o processo de definição da categoria linguística das palavras (classe morfológica), através do seu comportamento morfossintático. Como o texto é uma fonte não estruturada de informação, para torná-lo como uma entrada adequada para um método automático de extração de informação é necessário transformá-lo num formato estruturado (Patheja, Waoo, & Garg, 2012).

1.3 *Text Mining* e Análise de Sentimentos

Segundo Gharehchopogh & Khalifelu (2011), *Text Mining* é definido como o processo de analisar texto, com o objetivo de extrair informação útil a determinado propósito sendo o mesmo um processo complicado de se lidar tendo em conta a falta de estruturação dos dados. Essa estruturação existe em *Data Mining*. Alguns autores consideram que os

números definem *Data Mining* e o texto define *Text Mining*, sendo esta a grande diferença entre os dois conceitos: o primeiro necessita de dados estruturados para serem analisados e conseguir-se as respostas pretendidas; o segundo, *Text Mining*, debruça-se sobre dados não estruturados retirando significado dos mesmos.

A Análise de Sentimentos, ou *Text Mining*, é definida por Bing (2012) como a área de estudo que analisa as opiniões, sentimentos, avaliações e atitudes das pessoas com relação a diferentes entidades que podem ser produtos, serviços, organizações, indivíduos, eventos, tópicos, problemas e seus respetivos atributos.

Segundo o dicionário online "https://www.infopedia.pt/dicionarios/lingua-portuguesa" a etimologia da palavra "sentimento" expressa, entre outros, o "ato ou efeito de sentir; estado afetivo que tem por antecedente imediato uma representação mental; paixão; emoção; entusiasmo; opinião; ponto de vista; convicção; ideia". O significado aqui atribuído é coincidente com a de Pang (2006). No entanto, este autor reparte o seu significado em dois grupos:

- sentimento = atitude, pensamento ou julgamento; predileção, visão específica ou a uma determinada noção/parecer, opinião;
- sentimento = emoção, um sentimento refinado, idealismo emocional; sentimentalismo.

Também Indurkhya & Damerau (2010) classificam a informação textual em dois tipos principais:

- Factos são expressões objetivas sobre entidades, eventos e as suas propriedades;
- Opiniões são expressões subjetivas que descrevem os sentimentos da população, avaliações ou sentimentos acerca das entidades, e das suas propriedades.

A realização deste trabalho entronca na ideia de 'sentimento' relacionado com o conceito de 'opinião' enquanto base dos valores pessoais de um indivíduo, os seus sentimentos e emoções. A existência dessa característica permite classificar documentos textuais em dois grupos (Liu, 2010).

Segundo Liu & Zhang (2012), Análise de Sentimentos ou *Text Mining* é o estudo, por parte de um computador, de opiniões, atitudes, emoções ou mesmo preocupações de uma pessoa. Este conceito é de elevado interesse quer para organizações, que podem querer

saber por exemplo, o que os seus consumidores comentam sobre os seus produtos, quer para indivíduos singulares que, por vezes, necessitam de opiniões para tomar decisões.

Em Lima (2013) acentua-se que, apesar da Análise de Sentimentos ser apresentada em grande parte da literatura como estudo computacional de sentimentos, a mesma pode ser utilizada por muitos outros projetos. Uma vez que na Análise de Sentimentos se tratam problemas de classificação, estas técnicas podem ser utilizadas na classificação de documentos de texto, de acordo com a sua polaridade, ainda que do texto não se denote algum sentimento. Em Liu (2011) "opinião" é definida como um sentimento, emoção e/ou ponto de vista sobre uma entidade ou uma característica dessa entidade dada por um emissor, representado pela quíntupla:

$$(e_j, f_{jk}, so_{ijkl}, h_i, t_l)$$

Onde:

- ej: Entidade j que a opinião é direcionada.
- fjk: Característica k da entidade ej no qual é o foco da opinião.
- soijkl: valor (ou orientação) sentimental i sobre a característica fjk da entidade ej.
- hi: Emissor i da opinião.
- tl: Momento l em que a opinião foi expressa

A Análise de Sentimentos é uma área da computação que tem como o objetivo estudar esses tipos de informações presentes em textos (Pan, 2012) e é geralmente utilizada como estrutura base a quíntupla apresentada em Liu (2011).

A classificação de textos envolve normalmente duas tarefas (Liu, 2010):

- classificar uma frase como objetiva ou subjetiva;
- classificar as frases subjetivas (opiniões expressas) como negativas, positivas ou neutras

As temáticas relacionadas com a extração automática de opiniões/sentimentos em textos têm vindo a ser alvo de diversos desenvolvimentos nos últimos anos. A abordagem dominante para este problema é baseada em técnicas de aprendizagem de máquina: um processo indutivo que cria automaticamente um classificador aprendendo as características das classes/categorias, num conjunto de documentos pré-classificados (Sebastiani, 2002).

1.4 Aprendizagem-máquina

Aprender é o ato de adquirir conhecimento ou domínio (de assunto, matéria, etc.) através do estudo ou da prática ("https://www.infopedia.pt/dicionarios/lingua-portuguesa"). A aquisição de conhecimento pode ser concretizada de três formas: indução, dedução ou abdução. No método dedutivo o conhecimento é obtido a partir do geral até ao particular. No método indutivo, o conhecimento é formado através das partes até atingir a generalização. Já no método abdutivo usa-se a conclusão e a regra para defender que a premissa pode explicar a conclusão.

O raciocínio indutivo é largamente referido em teorias de aprendizagem-máquina. Tal como vem referido em Mitchell (1997): "um programa de computador aprende a partir de uma experiência E em relação a algumas classes de tarefa T e medidas de desempenho P, se esse desempenho na tarefa T, conforme medido por P, melhora a experiência E". São estes três elementos (experiência, tarefa e medida de desempenho) que suportam o processo de aprendizagem-máquina. Assim, a definição de aprendizagem-máquina relaciona-se com a capacidade de uma máquina mudar o seu funcionamento para melhorar seu desempenho, tendo como base para essas ações as entradas do sistema ou alguma influência externa (Nilsson, 1998).

O processo de classificação de sentimentos com base no seu "valor sentimental" consiste basicamente em agrupar as opiniões em duas classes: documentos com opiniões positivas e documentos com opiniões negativas. Para esse efeito, é criado um modelo que consiga identificar a classe a que o documento pertence. Os trabalhos realizados por Pang, Lee, & Vaithyanathan (2002) concretizam a aplicação de técnicas de aprendizagem-máquina neste domínio.

De acordo com Russell & Norvig (2010), a aprendizagem-máquina pode ser supervisionada, não supervisionada ou de reforço. Utilizamos a aprendizagem supervisionada quando tentamos prever o valor de uma variável dependente a partir de uma lista de variáveis independentes. A aprendizagem não supervisionada é utilizada quando pretendemos obter uma representação mais informativa dos dados disponíveis e, por último, na aprendizagem de reforço a máquina procura "aprender" qual a melhor ação a ser tomada de modo a maximizar o resultado Não comentaremos a abordagem utilizada neste último tipo de aprendizagem-máquina uma vez que não a utilizaremos nos trabalhos exploratórios.

1.4.1 Aprendizagem não supervisionada

Na aprendizagem não supervisionada não há conhecimento adequado de como os dados se comportam e são utilizados algoritmos que tentam organizá-los de modo a classificá-los. Segundo Khan, Daud, Nasir, & Amjad (2016), na aprendizagem de máquina não supervisionada, o modelo não é treinado. A tarefa é alcançada encontrando intra-semelhança e inter-similaridade entre objetos. A abordagem mais comum da categoria não supervisionada é o agrupamento (*clustering*).

Na aprendizagem não supervisionada podemos derivar a estrutura dos dados, agrupando-os com base em relações entre as variáveis. Também pode ser usada para reduzir o número de dimensões em um conjunto de dados para concentrar somente nos atributos mais úteis, ou para detetar tendências.

A utilização de técnicas aprendizagem não supervisionada neste trabalho ficou restringida à análise *n-grams* e *POS-Tagging*, cujas especificações e modo de funcionamento serão devidamente explicitadas no Capítulo 2.

1.4.2 Aprendizagem supervisionada

A classificação de textos tem como finalidade a sua atribuição/vínculo a uma determinada classe previamente estabelecida. Nos algoritmos de classificação utilizados em *Text Mining*, a classificação de cada documento é normalmente efetuada a partir dos termos, ou conjuntos de termos nele incluídos, tendo por base uma amostra do "universo" de documentos disponíveis previamente classificados e usados pelo investigador.

Nas técnicas de aprendizagem supervisionada é utilizado um conjunto de dados rotulados para treino do classificador e este "aprenderá" a identificar valores sentimentais com base nas características utilizadas nessa tarefa. Na fase de treino, o supervisor vai ajustando a aprendizagem indicando as saídas corretas, a partir do conjunto de treino com os dados previamente rotulados. Em Sebastiani (2002) constrói-se uma função classificadora para o treino supervisionado definida por:

$$f: D \times C \rightarrow \{0,1\}$$

Com:

- D - coleção de elementos a serem classificados;
- C - categorias/classes existentes.

A função descrita relaciona os dois conjuntos e atribui ao resultado o valor 1 (um) quando o documento pertence a uma classe, e o valor 0 (zero) quando o documento não pertence a essa classe.

1.4.2.1 Classificação com Naive Bayes

O raciocínio *Bayes*iano fornece uma abordagem probabilística para a inferência. É baseado na suposição de que as quantidades em estudo são governadas por distribuições de probabilidade e que podem ser tomadas decisões ótimas com base no raciocínio sobre essas probabilidades juntamente com os dados observados (Mitchell, 1997).

O *Naive Bayes* é um algoritmo que se tornou popular na área de aprendizagem de máquina ("*Machine Learning*") para categorizar textos baseado na frequência das palavras usadas. Trata-se de um classificador suportado no teorema de *Bayes* e é frequentemente utilizado na classificação de textos por ser rápido e fácil de implementar (Rennie, Shih, Teevan, & Karger, 2003). Em Lima (2013) refere-se que o classificador *Naive Bayes* é considerado um dos mais eficientes em questões relacionadas com processamento e precisão na classificação de novas amostras.

$$P(A|B) = P(B|A) \times P(A) / P(B)$$

O teorema de *Bayes* representado na fórmula anterior foi desenvolvido por Thomas *Bayes* em meados do século XVIII e é normalmente designado por fórmula de probabilidade condicional de um determinado evento. Sendo B um evento passado e A um evento que depende de B, a probabilidade posterior da ocorrência do evento A dado o evento B, designada por P(A|B), é calculada através da contagem dos casos em que A e B ocorrem juntos dividida pelo número de casos em que apenas ocorre o evento B.

1.4.2.2 Classificação com LexiconPT

Uma outra técnica de classificação é a baseada em recursos léxicos, também chamada de baseada em dicionários ou linguística e tem como foco o emprego de dicionários de palavras e/ou expressões que possuem classes e valores pré-definidos. A técnica mais utilizada com recursos léxicos é a da ocorrência de sentimento próximo a uma característica da entidade. Esta técnica possui bons resultados quando empregada em sentenças ou textos pequenos, pois a proximidade com a característica é menor. Quando a análise é feita em textos maiores, as características podem estar numa frase e a opinião noutra, o que pode reduzir a precisão da técnica (Hu & Liu, 2004).

Os léxicos são bases de dados criadas manualmente (ou automatizada) e que tem como objetivo serem usadas como base de conhecimento. Normalmente os léxicos são direcionados a uma área específica. Nos léxicos, as palavras têm já as respetivas polaridades associadas. Isso significa que temos já o sentimento de cada palavra nessa base de dados. No entanto, existem outros fatores a serem considerados nesta abordagem, como é o caso das palavras de negação e de inversão de valores em que se expressa a negação de um sentimento negativo.

A vantagem dos léxicos é que não é necessário rotular os dados para treino. A maior desvantagem é que os léxicos são limitados a um idioma e ao tamanho da base de dados. O "SentilexPT" é um léxico para o idioma português que está disponível para pesquisa e desenvolvimento. Possui 6.531 adjetivos com informações de polaridade, alvo do sentimento e método de atribuição de polaridade.

1.4.2.3 Classificação com Support Vector Machine (SVM)

Na classificação de sentimentos expressos em textos são também frequentemente utilizadas técnicas de aprendizagem-máquina suportadas em algoritmos SVM. As SVMs são suportadas pela teoria da aprendizagem estatística desenvolvida por Vapnik (1998). Também estes algoritmos classificam os dados em dois grupos (positivo e negativo). Durante o treino, o SVM encontrará um hiperplano que consegue separar (o máximo possível) os elementos entre as classes de valor sentimental positivo e negativo. O resultado do processo de treino de uma SVM é um modelo para classificação de sentimentos expressos num determinado texto/documento.

O SVM utiliza o hiperplano ou fronteira que melhor separa os casos positivos dos negativos num espaço n-dimensional. Essa fronteira é a que tem a distância máxima (a partir do vetor de suporte mais próximo) entre as superfícies por ela separadas. As características dos elementos a classificar pelo SVM são representadas por uma dimensão desse espaço n-dimensional. Este mesmo hiperplano é apresentado por Lorena (2006) como:

$$f(x) = w \cdot x + b \ = 0 \ ,$$

Com:

- w - vetor pertencente ao conjunto de entradas;

- x - um ponto no hiperplano;
- b - valor da bias, isto é, o valor assumido como resultado caso não exista nenhuma entrada que influencie a aprendizagem.

Uma das tarefas no qual o SVM tem bons resultados é na classificação de texto e isso acontece devido às seguintes características (Joachims, 1998):

- Espaço de entrada de alta dimensão com possibilidade de utilização de um elevado número de características classificadoras. O número de características (dimensões) não tem um impacto significativo na construção do hiperplano.
- Poucas características irrelevantes. Não há limite para a quantidade de características. A este propósito, o trabalho do pesquisador consiste em identificar as características irrelevantes para o estudo.

Os estudos realizados por Kivinen, Warmuth, & Auer (1997) descrevem experiências com utilização destes algoritmos, concluindo que são apropriados para tratamento de documentos densos ou de documentos esparsos, não existindo, portanto, problemas de um treino excessivo nas SVM.

Como anteriormente se referiu, a maioria dos problemas de classificação de texto é linearmente separável em duas classes (ou grupos), e, sendo o SVM um classificador linear, então é também passível de ser utilizado no tratamento de dados compostos por várias características.

As SVMs lineares são eficazes na classificação de conjuntos de dados linearmente separáveis ou que possuam uma distribuição aproximadamente linear, sendo que a versão de margens suaves tolera a presença de alguns ruídos e *outliers*. Porém, há muitos casos em que não é possível dividir satisfatoriamente os dados de treino por um hiperplano (Lorena, 2006). Nestes casos, a solução passa pela utilização de um Kernel[5] devido à simplicidade do seu cálculo e à sua enorme capacidade na representação de espaços abstratos.

Na realização deste trabalho e para além dos SVMs lineares, testámos também o Kernel polinomial e o Kernel gaussiano (Radial Basis), ambos inspirados no teorema de Cover,

[5] Função que recebe dois pontos xi e xj do espaço de entradas e computa o produto escalar desses dados no espaço de características

segundo o qual, a um conjunto de dados de treino que não é linearmente separável, é possível transformá-lo, com alta probabilidade, num conjunto separável linearmente, projetando-o num espaço de maior dimensão através de uma transformação não linear. Para isso duas condições devem ser satisfeitas. A primeira é que a transformação seja não linear, enquanto a segunda é que a dimensão do espaço de características seja suficientemente alta (Lorena, 2006).

A classificação baseada nos algoritmos SVM e *Naive Bayes* apresentam, em geral, resultados precisos e confiáveis na classificação dos dados de texto não estruturados (Sulova, Todoranova, Penchev, & Nacheva, 2017).

1.5 Trabalhos relacionados

Nos últimos anos temos assistido à divulgação de inúmeras experiências que envolvem a utilização de *tweets* para Análise de Sentimentos recorrendo a vários métodos de classificação numa ampla gama de contextos.

Parikh & Movassate (2009) implementaram duas técnicas de classificação recorrendo ao classificador *Naive Bayes*, uma com modelo de *bigrams* e a outra com Entropia Máxima. Os resultados obtidos confirmam melhor desempenho com utilização do classificador simples (sem a utilização de Entropia Máxima), deixando, porém, uma ressalva quanto ao tamanho do corpus[6] utilizado de modo a validar a fidedignidade da metodologia.

Talbot, Acheampong, & Wicentowski (2015) descrevem um sistema de classificação de sentimentos que emprega uma abordagem de categorização de texto supervisionada e restrita. Na primeira parte do trabalho e uma vez que o pré-processamento completo dos dados de *tweets* se mostrou eficaz em tarefas anteriores de classificação de sentimentos, foram introduzidas várias etapas de pré-processamento para melhorar a qualidade da informação lexical. De seguida, foi utilizado um classificador *Naive Bayes* para detetar o sentimento do *tweet*. O classificador é treinado apenas nos dados de treino fornecidos pelos investigadores. O sistema utilizava listas externas geradas pelo investigador, de palavras positivas e negativas em várias etapas ao longo da classificação e produziu um F-score total de 59,26% no conjunto de testes.

[6] Coleção de documentos que contêm textos escritos em linguagem natural

Troussas, Virvou, Espinosa, Llaguno, & Caro (2013) utilizaram a rede social Facebook juntamente com o algoritmo *Naive Bayes* e apresentaram vários métodos de representação de dados, realizando melhorias significativas sobre modelos de *unigrams*, com o desenvolvimento de um sistema de aprendizagem de línguas. A recolha de dados foi realizada de forma aleatória e ficou demonstrada a elevada precisão do modelo gerado.

Gamallo, Garcia, & Fernandez-Lanza (2013) descreveram uma estratégia baseada num classificador *Naive Bayes* para detetar a polaridade em *tweets* espanhóis. As experiências mostraram que o melhor desempenho é obtido usando um classificador binário, distinguindo apenas duas categorias de polaridade aguda: positiva e negativa. Para identificar mais níveis de polaridade, o sistema é fornecido com limites definidos experimentalmente para detetar valores fortes, médios e fracos (ou neutros). Além disso, a fim de detetar *tweets* com e sem polaridade, o sistema faz uso de uma regra muito básica, que procura palavras de polaridade dentro do texto analisado. Os resultados da avaliação mostram um bom desempenho do modelo (cerca de 67% de precisão) quando é usado para detetar quatro categorias de sentimento.

No campo da administração pública surgiram algumas pesquisas sobre a utilização de redes sociais análise de sentimento expressos em redes sociais. Fortuny, Smedt, Martens, & Daelemans (2012) debruçaram-se sobre as questões políticas na Bélgica, num período de crise no final do ano de 2011, com o objetivo de medir o "sentimento social" dos cidadãos em relação aos partidos políticos do país, tendo por base os dados extraídos das notícias de grandes jornais em versões *online*.

Em Portugal, o trabalho realizado por Varela (2012) procura obter um sistema de classificação que seja independente da linguagem e realizou análises dos resultados através de bases de dados sobre cinema na língua espanhola e portuguesa (*Internet Movie Database* - IMDb). As experiências realizadas concluíram que o classificador Multinomial *Naive Bayes* é o que melhor se adequa à classificação de documentos mais pequenos.

Arunachalam & Sarkar (2013) propuseram um modelo e um estudo do caso para monitorizar e analisar o sentimento dos cidadãos nas redes sociais (Twitter, Facebook, YouTube, etc.) acerca das principais organizações que administram os apoios sociais nos Estados Unidos da América. No processo, os decisores obtiveram informações valiosas, que puderam ser convertidas em indicadores de gestão.

A utilização das redes sociais pelas administrações tributárias foi abordada por Dellia & Tjahyanto (2017), onde é estudada a utilização do Twitter para analisar as queixas/reclamações tributárias. O estudo revelou as dificuldades no tratamento automático de reclamações fiscais derivado da enorme quantidade de publicações não respeitantes à matéria em análise.

Os diversos estudos que tivemos oportunidade de analisar e que envolvem a utilização de *tweets* para análise e classificação de sentimentos neles expressos, manifestavam as dificuldades no tratamento do "ruído" e na análise semântica. Um outro problema citado por vários autores tem que ver com o facto de existirem poucas bibliotecas com disponibilização de radicais (*stem*) das palavras escritas em português, o que pode originar conjugações diferentes para uma mesma palavra e eventuais problemas de classificação.

2 METODOLOGIA

Nesta secção apresentamos uma descrição da metodologia utilizada no desenvolvimento deste trabalho, iniciando com uma apresentação resumida da metodologia de investigação "*Cross Industry Standard Process for Data Mining*" (CRISP-DM) e do método utilizado em *Text Mining*, Em seguida, descreveremos resumidamente a rede social Twitter e abordaremos as técnicas utilizadas na extração dos *tweets*. Terminamos esta seção com a apresentação do software e das técnicas auxiliares utilizadas no pré-tratamento dos *tweets*.

2.1 O modelo CRISP-DM

A metodologia padrão CRISP-DM é descrita como um modelo com processos hierárquicos que identifica as diferentes fases na implantação de um projeto de *Data Mining*, consistindo num conjunto de tarefas descritas em quatro níveis de abstração: fases do processo, tarefas genéricas de cada fase, as tarefas específicas e as instâncias do processo (Chapman, et al., 2000).

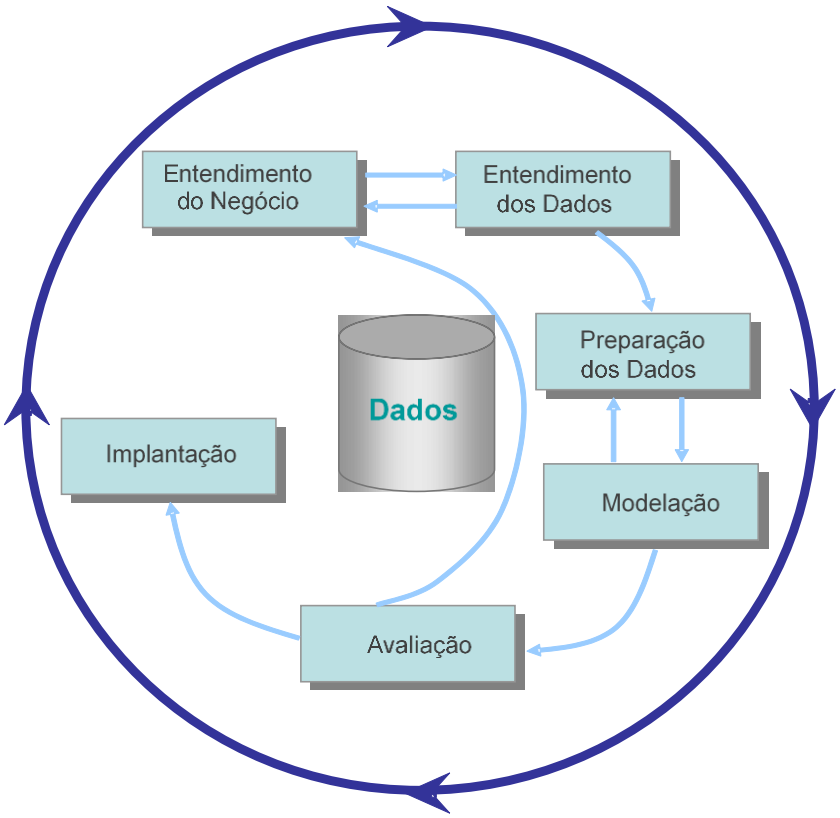


Figura 2.1 Modelo CRISP-DM

Fonte: Adaptado de Chapman et al. (2000)

A metodologia CRISP-DM apresenta-se como um modelo genérico a utilizar na implantação de um qualquer projeto de *Data Mining*. No entanto, o método utilizado em *Text Mining* tem características próprias que implicam o ajustamento da metodologia. Assim, as fases presentes na Figura 2.1 foram concebidas do seguinte modo:

- Conhecimento do negócio: os governos e a administração pública estão, hoje, bastante sensibilizados para enorme popularidade das redes sociais e já começaram a utilizá-las como forma de facilitação e iteratividade da sua comunicação com os cidadãos. O tratamento automatizado de informação pertinente constitui um precioso auxílio à tomada de decisões e para a melhoria da interação daqueles Serviços com os cidadãos;
- Conhecimento dos dados: durante a primeira parte do trabalho exploratório, visualizámos os dados obtidos no processo de extração de *tweets* com vista a conhecer a sua estrutura, principais características e informação relevante para a análise que nos propomos efetuar;
- Preparação dos dados: conhecida a estrutura e composição dos dados, na fase seguinte procedemos aos ajustamentos necessários para a concretização de cada uma das tarefas. Note-se que a preparação específica dos dados a utilizar num determinado modelo pode divergir, em pequenas nuances, da preparação dos dados a utilizar noutro modelo;
- Modelação dos dados: no nosso caso, as tarefas a realizar nesta fase consistem em selecionar cada uma das técnicas escolhidas e calibrar os respetivos parâmetros para valores suscetíveis de aprofundar o conhecimento sobre a matéria em análise;
- Avaliação dos resultados: esta fase do projeto consiste em avaliar os resultados da aplicação de cada uma das técnicas pré-definidas e compará-los entre si. A realidade mostrou-nos que o processo de desenvolvimento e aplicação de um determinado modelo teórico envolve, em muitos casos, um retrocesso nas fases da metodologia CRISP-DM, e um novo recomeço, agora com uma nova "visão" sobre o problema.
- Implantação: fase final em que se coloca em produção o modelo desenvolvido. No nosso caso o modelo desenvolvido não foi colocado ainda em produção, mas estamos esperançados e disponíveis para que, num futuro próximo, possamos

contribuir para a implantação do método proposto, ainda que a título experimental e/ou aplicado num contexto específico.

2.2 O método utilizado em *Text Mining*

O processo de extração de informação em textos não estruturados é normalmente decomposto em 4 fases (adaptado de Liu e Zhang, 2012):

- Identificação do problema – esta fase assemelha-se, em muito, à primeira fase da metodologia CRISP-DM. É nesta fase que se fixa o âmbito, se identificam as fontes de informação disponíveis, escolhem-se as metodologias de tratamento da informação e se definem os objetivos a alcançar durante o processo;
- Pré-processamento – os processos de *Text Mining* envolvem a execução de diversas tarefas de tratamento e padronização para posterior transformação em formato estruturado. No final dessa fase será possível transformar a coleção de textos numa representação estruturada em formato tabela atributo-valor;
- Extração de padrões – na fase seguinte procuram-se padrões com recurso a técnicas de agrupamento utilizando a medidas de proximidade.
- Pós-processamento e validação dos resultados – fase de validação dos resultados obtidos recorrendo a indicadores estatísticos.

Como atrás ficou referido, o presente trabalho tem como objetivo geral, a aplicação de técnicas de Análise de Sentimentos a um conjunto de *posts* publicados na rede social Twitter que expressam opiniões relacionadas com o funcionamento geral das autoridades fiscais portuguesas, bem como, avaliar a possibilidade de implantação de um modelo para acompanhamento e monitorização dessas manifestações de opinião.

2.3 Extração dos tweets

O Twitter é uma rede social com mais de 600 milhões de utilizadores e permite a *postagem* de pequenas mensagens de texto (limitadas a 140 caracteres), conhecidas como *tweets*. Qualquer utilizador tem acesso às informações permanentemente colocadas na rede, sem quaisquer restrições na permissão de conexão entre eles. Para Russell (2013) essa é a sua grande característica diferenciadora das demais redes sociais mais populares, como o LinkedIn e o Facebook, e que faz com que o Twitter seja uma rede social com maior potencial de exploração.

Na rede social Twitter a partilha informações (*tweets*) é efetuada sob a forma de textos, *links*[7] e *hashtags*[8] publicados na rede e que podem ser visualizados por todos os seguidores do autor do *tweet*. Existe a possibilidade desses seguidores responderem ao *tweet* ou partilharem-no com os seus próprios seguidores (*retweet*). Para poder recolher os *tweets* num formato estruturado é necessário recorrer às credenciais de acesso aos dados da *Application Programming Interface* (API) disponibilizada pelo Twitter.

As credenciais são obtidas através da subscrição de uma conta de "desenvolvedor" do Twitter. Limitámos a extração a um conjunto de palavras-chave (Autoridade Tributária, Autoridades Fiscais, Finanças, Fisco, IVA, IRS e IRC) que se relacionam diretamente com o tema em análise e que foram escritas em língua portuguesa Devido à limitação temporal imposta pela aplicação de recolha, não é possível extrair *tweets* com uma data de criação anterior a 10 dias em relação à data de extração.

A fase de extração dos *tweets* decorreu entre agosto de 2018 e fevereiro de 2019 (7 meses) e as respetivas datas de criação na rede correspondem aproximadamente ao mesmo período. Ultrapassada a fase de identificação do problema e de recolha dos dados, passámos à fase de pré-processamento, executando diversas tarefas de tratamento e padronização, seleção de termos relevantes e posterior transformação em formato estruturado para facilitação da extração de padrões. O reflexo da abrangência do tema em estudo na quantidade de *tweets* recolhidos, a par das limitações nele impostas quanto ao seu alcance, implicou o recurso a diversas técnicas de *Text Mining*, de modo a restringir o universo dos *tweets* associados ao tema em análise.

2.4 O software utilizado

Para além da extração, utilizámos também o software RStudio[9], versão 3.5.2, na exploração do conteúdo dos *tweets*, por se tratar de um software gratuito e que contém vários recursos disponíveis para análise de textos. Ao carregar o software R são imediatamente instaladas diversas funções básicas necessárias para o seu funcionamento.

[7] Ligação (em inglês, link), é uma referência dentro de um documento em hipertexto a outras partes desse documento ou a outro documento.
[8] Palavras-chave (relevantes) ou termos associados a uma informação, tópico ou discussão que se deseja indexar de forma explícita no aplicativo Twitter, antecedidas pelo símbolo cardinal (#).
[9] RStudio é um software livre de ambiente de desenvolvimento integrado para R, uma linguagem de programação para gráficos e cálculos estatísticos (https://www.rstudio.com/)

No entanto, está disponível uma vasta gama de bibliotecas (pacotes) com aplicação em projetos de *Text Mining*.

A biblioteca 'tm' do "R" possui as principais funções utilizadas em *Text Mining*, sendo o mais utilizado no tratamento e análise de textos que inclui diversas funções com um uma estrutura pré-definida (ex.: corpus, tm_map, stopwords, termFreq, etc). O dicionário "readxl" é indispensável para a importação de ficheiros "*Excel"* (.xlsx). Para além destas, utilizámos também outras bibliotecas cujas funções se explicitarão à medida da sua implementação.

Na extração dos *tweets* utilizámos a API de pesquisa disponibilizada pela rede Twitter, recorrendo a um conjunto de instruções em linguagem R para extração dos *tweets* em que estivessem presentes os termos (ou conjugações dos termos) "Autoridade Tributaria", "Autoridades Fiscais", "Finanças", "Fisco", "IRS", "IVA" ou "IRC". Os termos escolhidos para a pesquisa são aqueles que, na opinião do autor, melhor se identificam com o problema e que têm maior probabilidade de estarem incluídos em comentários acerca do tema em análise. A Figura 2.2 discrimina o conjunto de instruções, em linguagem "R", para extração periódica dos *tweets*.

```
library(twitteR)
api_key <- "xxxxxxxxxxxxxxxxxxxxxxxx"
api_secret <- "yyyyyyyyyyyyyyyyyyyyyyyyyyyyyyyy"
token <- "zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz"
token_secret <- "wwwwwwwwwwwwwwwwwwwww"
setup_twitter_oauth(api_key, api_secret, token, token_secret)
tweets <- searchTwitter("(Autoridade AND Tributaria) OR
(#Autoridade AND Tributaria) OR (Autoridades AND Fiscais) OR
(#Autoridades AND Fiscais) OR Finanças OR Fisco OR #Finanças
OR #Fisco OR IRS OR #IRS OR IVA OR #IVA OR IRC OR #IRC", n =
20000, lang = "pt")
```

Figura 2.2 Script em "R" para extração de tweets

A limitação temporal imposta pela rede social Twitter na extração de *tweets*, que impossibilita a extração de *tweets* com uma data de criação anterior a 10 dias em relação à data de extração, obrigou a uma execução periódica do conjunto de instruções "R" acima referido, exportando os resultados de cada extração para um ficheiro do tipo CSV. No final do período programado para a extração de *tweets*, juntámos os registos de todos os ficheiros do tipo "csv" num único ficheiro, que designámos por "teste1.csv" e que contém o conjunto total dos dados recolhidos para análise.

2.5 Técnicas auxiliares para pré-tratamento dos tweets

Durante a análise do ficheiro criado na etapa anterior (teste1 = twt1) verificámos que este continha 84.664 linhas correspondentes a outros tantos *tweets* inseridos na rede social Twitter entre 2018-07-29 e 2019-02-25, dos quais retirámos apenas os campos com informação pertinente:

- doc_id, correspondente ao número interno de identificação do *tweet*;
- text, correspondente ao texto inserido pelo utilizador;
- created, correspondente à data de inserção do *tweet* na rede social;
- screenName, correspondente ao nome interno do utilizador do Twitter;
- retweetCount, correspondente à contagem de *retweets* do *tweet*.

Através de um conjunto de instruções destinadas a eliminar os registos não relacionados com a temática em estudo, discriminadas no Anexo 1 deste relatório, reunimos um conjunto de registos selecionados para a fase de pré-processamento corresponde a cerca de 30% da quantidade inicial de *tweets*. Esta tarefa foi realizada com recurso às técnicas de contagem simples de termos mais frequentes. Efetuámos também algumas correções de erros ortográficos no texto em palavras com elevada frequência.

A segunda fase do pré-processamento envolveu o recurso à análise *n-grams*[10], tendo em vista a exclusão de frases fora do contexto em análise. Para aplicação desta técnica é necessário agrupar os *tweets* repetidos num único registo (texto) que integrarão o *corpus*. O conjunto de *tweets* (sem repetições) que correspondem ao *corpus* é composto por 12.663 registos.

O passo seguinte consistiu na criação de um script em "R" que executa as transformações necessárias à criação do *corpus*, agora num formato compatível para aplicação de técnicas baseadas na análise *n-grams*. Nesta fase, e depois de convertida toda a coleção de *tweets* num único *corpus*, são removidas as *stopwords*[11], a pontuação e os espaços em branco extra existentes no *corpus*, para além da transformação das palavras em minúsculas.

[10] Sequência contígua de n itens de uma dada amostra de texto
[11] Palavras com alta frequência em textos e que normalmente não acrescentam conteúdo, tais como preposições, artigos, conjunções e outros.

Apoiados nas técnicas que suportam a análise *n-grams*, listámos 300 sequências de duas palavras que ocorrem simultaneamente. O *output* gerado possibilita a eliminação de *tweets* que incluem *bigrams* que indiciam não serem relacionados com o objetivo deste estudo. Ao *dataset* resultante aplicámos o mesmo algoritmo, fazendo agora corresponder n=3. Em seguida, aplicámos a mesma técnica, fazendo corresponder n=2 e calculando a medida "*Term Frequency – Inverse Document Frequency*" (TF-IDF) para os pares de termos (Salton & Buckley, 1988).

O valor TF-IDF é uma medida estatística que indica a importância de uma palavra de presente num documento em relação a uma coleção de documentos. O seu valor aumenta proporcionalmente à medida que aumenta o número de ocorrências de uma palavra num documento, no entanto, esse valor é equilibrado pela frequência dessa palavra no *corpus*:

- TF – o peso de um termo que ocorre num documento é diretamente proporcional à sua frequência;
- IDF – a especificidade de um termo pode ser quantificada por uma função inversa do número de documentos em que ele ocorre.

Estes procedimentos permitiram a exclusão de 647 *tweets* não relacionados com a temática em estudo. No final desta etapa obtivemos um *dataset* com 12.016 registos correspondentes a outros tantos *tweets* (sem repetições).

Recorrendo às técnicas de etiquetagem morfossintática ("*POS-Tagging*"), procedemos à exclusão de frases que não estão escritas na língua portuguesa. Para além das bibliotecas anteriormente descritas, estas técnicas são apoiadas em funções disponíveis nas bibliotecas "ptstem", "udpipe" e "textcat".

A biblioteca "ptstem" engloba uma coleção de algoritmos de *stemming*[12] para a língua portuguesa. Já a biblioteca "udpipe" consiste num kit de ferramentas de processamento de linguagem natural que fornece funções de 'tokenização' independente do idioma, '*POS-Tagging*, 'lematização' e 'análise de dependência' de texto. Por último, a biblioteca "textcat". Inclui algoritmos necessários para a categorização de textos com base na técnica *n-grams*.

[12] Processo de reduzir palavras flexionadas (ou às vezes derivadas) ao seu tronco (stem), base ou raiz.

Com a aplicação de um conjunto de instruções "R" carregámos o modelo de anotação de texto para a língua portuguesa ("portuguese-bosque-ud-2.3-181115.udpipe", disponível através da biblioteca "udpipe") e aplicámo-lo ao conjunto de dados obtido na etapa anterior. O conjunto de dados assim gerado inclui 11.464 registos, com todas as palavras incluídas nos *tweets* anotadas de acordo com a sua categoria linguística, tendo sido eliminados 552 tweets que o algoritmo identificou como não redigidos na língua portuguesa e que comprovámos por observação direta.

Efetuámos, também, a classificação manual de uma amostra aleatória de 1.015 casos, utilizada nos testes dos modelos selecionados e que adiante (subsecção 3.2.1) melhor se descreverá.

A Figura 2.3 resume o processo atrás descrito, de extração de *tweets*, pré-preparação e pré-processamento, para criação do conjunto de dados a utilizar, quer na análise *n-grams* e *POS-Tagging* (capítulo 3), quer na extração da amostra utilizada nos testes com utilização dos algoritmos "*Naive Bayes*" e "SVM", bem como da biblioteca "R" "Lexicon-PT".

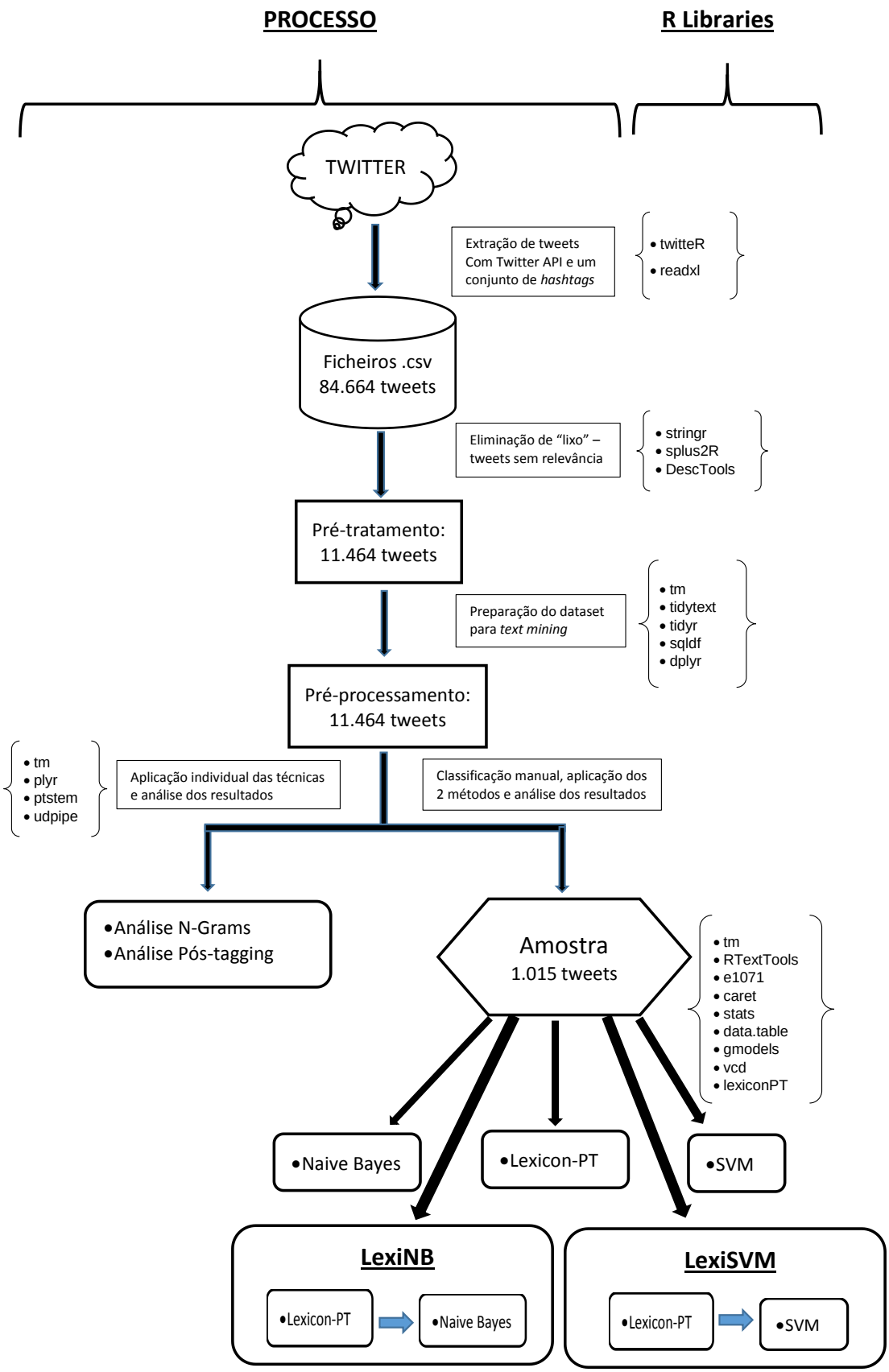


Figura 2.3 Resumo da metodologia utilizada

3 ANÁLISE E DISCUSSÃO DOS RESULTADOS

Neste capítulo efetuamos a descrição dos procedimentos de pré-processamento e apresentação dos resultados da análise sintática aplicada à globalidade dos *tweets* resultantes do pré-tratamento, bem como para a descrição das tarefas realizadas para a criação da amostra e de pré-processamento utilizados na geração dos modelos testados, finalizando com a discussão dos respetivos resultados.

3.1 Análise sintática

O recurso às análises *n-grams* e *POS-Tagging* em trabalhos desenvolvidos durante a fase de pré-processamento (ver Figura 2.3) permitiu excluir um conjunto de 1.204 *tweets* não relacionados com a temática em estudo. O processo de extração de características foi direcionado no sentido de identificar os termos associados às palavras-chave utilizadas na pesquisa, tendo em vista a obtenção das características mais discriminantes para a classificação a efetuar.

3.1.1 Análise *n-grams*

A abordagem *n-grams* (ver Anexo 2) permite capturar a estrutura da linguagem do ponto de vista estatístico. A técnica consiste na criação de um *corpus* a partir dos *tweets* selecionados, mapeado e sob a forma de uma tabela de dupla entrada em que cada linha corresponde ao documento (doc_id), as colunas correspondem às palavras (*tokens*) incluídas em cada documento e a cada par (linha/coluna) é atribuído o valor 1 ou 0 consoante o *token* esteja presente ou não no documento. Tendo em vista a análise evolutiva dos termos mais frequentes, efetuámos uma repartição do *dataset* final (11.464 registos) em 3 conjuntos:

- dt201809 – inclui os *tweets* cuja data de criação ocorreu entre 2018-07-29 e 2018-09-30;
- dt201812 - inclui os tweets cuja data de criação ocorreu entre 2018-10-01 e 2018-12-31;
- dt201902 - inclui os tweets cuja data de criação ocorreu entre 2019-01-01 e 2019-02-25.

Na Figura 3.1 apresentamos os resultados para os termos mais frequentes em cada um dos períodos:

	Unigram201809	n	perct		Unigram201812	n	perct		Unigram201902	n	perct
1	iva	4152	0.0574	1	iva	4891	0.0539	1	iva	4461	0.0567
2	é	1715	0.0237	2	é	2290	0.0252	2	é	1934	0.0246
3	fisco	1383	0.0191	3	fisco	2058	0.0227	3	fisco	1607	0.0204
4	irs	1357	0.0188	4	irs	1687	0.0186	4	irs	1425	0.0181
5	touradas	825	0.0114	5	touradas	862	0.00950	5	touradas	842	0.0107
6	6	553	0.00764	6	6	587	0.00647	6	6	563	0.00716
7	vai	441	0.00609	7	irc	542	0.00598	7	vai	467	0.00594
8	irc	407	0.00562	8	vai	520	0.00573	8	irc	455	0.00579
9	sobre	383	0.00529	9	sobre	459	0.00506	9	sobre	406	0.00516
10	ser	342	0.00473	10	ser	433	0.00477	10	ser	370	0.00471
11	redução	311	0.00430	11	pagar	358	0.00395	11	pagar	316	0.00402
12	ps	304	0.00420	12	redução	335	0.00369	12	redução	316	0.00402
13	baixar	300	0.00415	13	baixar	323	0.00356	13	ps	306	0.00389
14	pagar	282	0.00390	14	ter	320	0.00353	14	baixar	304	0.00387
15	eletricidade	274	0.00379	15	ps	308	0.00340	15	eletricidade	275	0.00350
16	governo	263	0.00363	16	23	300	0.00331	16	23	274	0.00348
17	23	249	0.00344	17	governo	296	0.00326	17	ter	274	0.00348
18	ter	246	0.00340	18	eletricidade	283	0.00312	18	governo	270	0.00343
19	taxa	242	0.00334	19	taxa	276	0.00304	19	taxa	258	0.00328

Figura 3.1 Análise evolutiva dos termos mais frequentes (output "R")

Com a utilização de *bigrams* (2 termos adjacentes) também não se registam alterações significativas nos termos mais frequentes para cada um dos diferentes períodos considerados na análise. De facto, o período temporal de extração de *tweets* é muito curto (7 meses), em que os temas dominantes estão relacionados com propostas para o Orçamento de Estado de 2019 para redução da taxa de IVA a aplicar nos ingressos em espetáculos tauromáquicos e nos consumos de eletricidade, para além da redução das taxas de IRS a aplicar a ex-emigrantes.

3.1.2 Part-of-speech Tagging (etiquetagem morfossintática)

O objetivo da etiquetagem morfossintática (*POS-Tagging*) é de classificar os *tokens* de uma frase (ou texto) de acordo com suas classes morfológicas (substantivo, verbo, adjetivo, etc.). Ao *dataset* final (11.464 registos), e para além da biblioteca "udpipe", necessária para a realização das tarefas de tokenização, *stemming*[13], lematização[14] e *POS-Tagging* de forma automática, utilizámos a biblioteca "lattice"[15] para visualização dos resultados desta análise.

[13] Os algoritmos de stemming funcionam cortando o fim ou o início da palavra, levando em conta uma lista de prefixos e sufixos comuns que podem ser encontrados numa palavra.
[14] processo de agrupar as formas flexionadas de uma palavra para que elas possam ser analisadas como um único item, identificado pelo lema da palavra ou pela forma de dicionário.
[15] Sistema de visualização de dados de alto nível, com ênfase em dados multivariados.

Durante os trabalhos exploratórios (ver Anexo 3) verificámos que o algoritmo não conseguia distinguir claramente a classe gramatical de alguns dos elementos do texto, como é o caso dos termos "fiscais", "fisco", "fiscal", "tributária" e outros, que, em determinado contexto poderão ter a categoria de "adjetivo", mas que, no tema em análise, terão a classe gramatical de "nome". Por outro lado, os termos "redução" e "descida" tinham também a classe gramatical de "nome" quando deveriam ter a classe gramatical de "verbo". Verificámos também que o *token* "iva" podia assumir as classes gramaticais de "nome", "verbo", ou "adjetivo", tendo-lhe sido atribuída, no contexto deste trabalho, a classe gramatical de "nome". No gráfico seguinte (Figura 3.2) apresentamos um resumo da distribuição morfossintática do texto após os ajustes referidos.

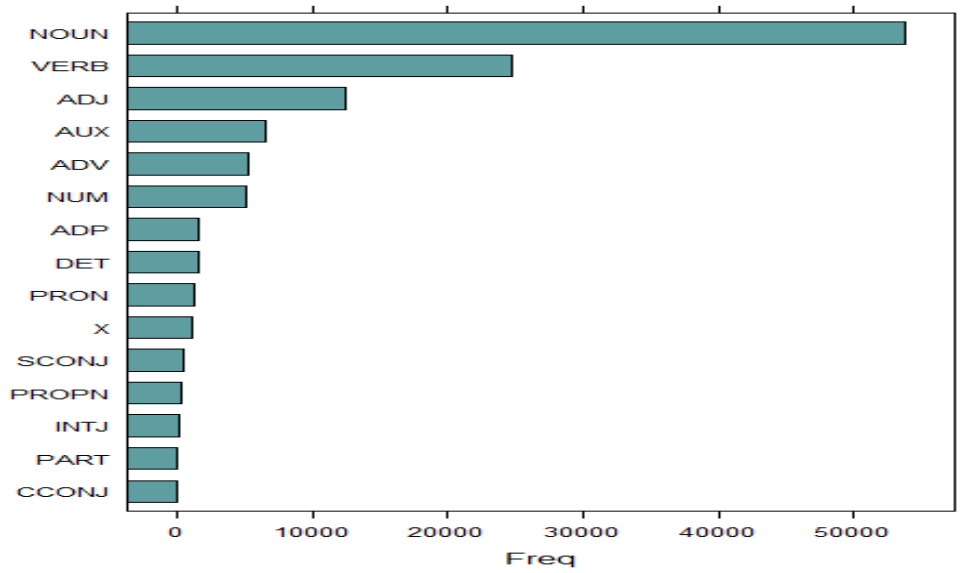


Figura 3.2 Frequências das classes gramaticais

A quantidade de "nomes" presentes nos *tweets* selecionados representa cerca de 48% da totalidade dos *tokens* existentes, enquanto os verbos têm um peso a rondar os 22%. Já os adjetivos representam apenas cerca de 9% da totalidade dos *tokens* e os verbos auxiliares e advérbios representam cerca de 5% cada.

A Figura 3.3 apresenta os nomes com maior número de ocorrências. Nesta análise, optámos por retirar as *hashtags* incluídas na pesquisa inicial de *tweets*, devido à sua expectável relevância. Os resultados confirmam a tendência observada na análise *n-grams*.

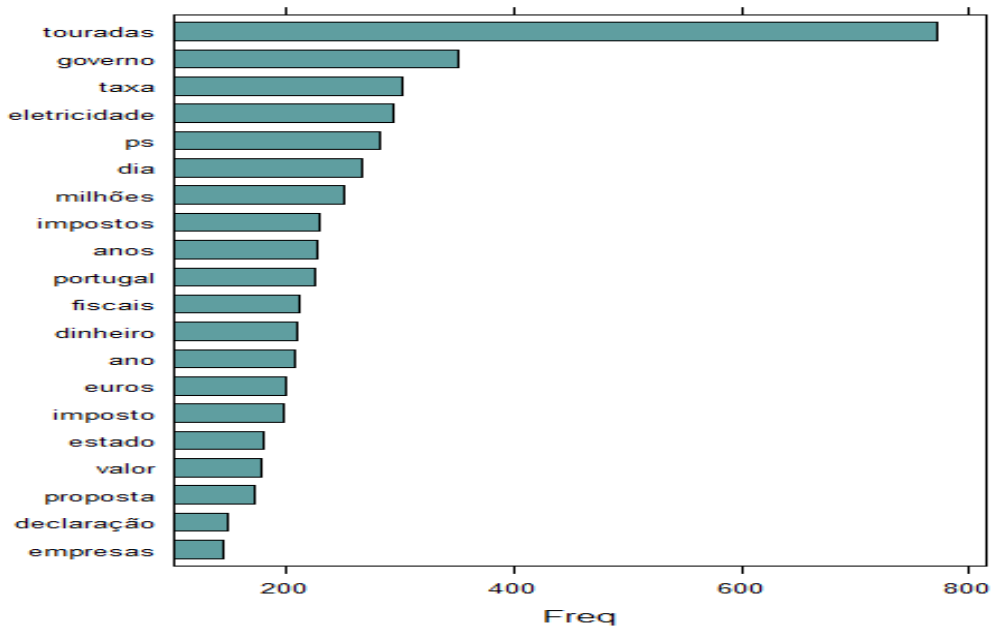


Figura 3.3 Frequências de termos da classe gramatical "nomes"

Na análise efetuada ao conjunto de verbos com maior relevância no conjunto de registos, observámos a ocorrência de verbos diretamente relacionados com os impostos, como é o caso dos verbos "pagar" e baixar". Efetuámos a mesma análise para a classe gramatical dos adjetivos, surgindo os termos "bom", "novo" e "maior" com maior predominância nesta categoria.

3.1.3 Análise de coocorrências

Recorrendo às funções disponíveis nas bibliotecas "tm" e "igraph" e outras, procedemos à análise de coocorrências, com várias simulações para pares "nome – adjetivo", "nome – verbo" e "verbo – adjetivo". O processo implica a criação de um *corpus* mapeado, em que o grau de coocorrência é medido por uma função de verosimilhança (Log-likelihood). A representação gráfica desta medida reflete-se na dimensão dos círculos de cor amarelada. Na Figura 3.4 apresentamos os resultados deste método para o termo "iva".

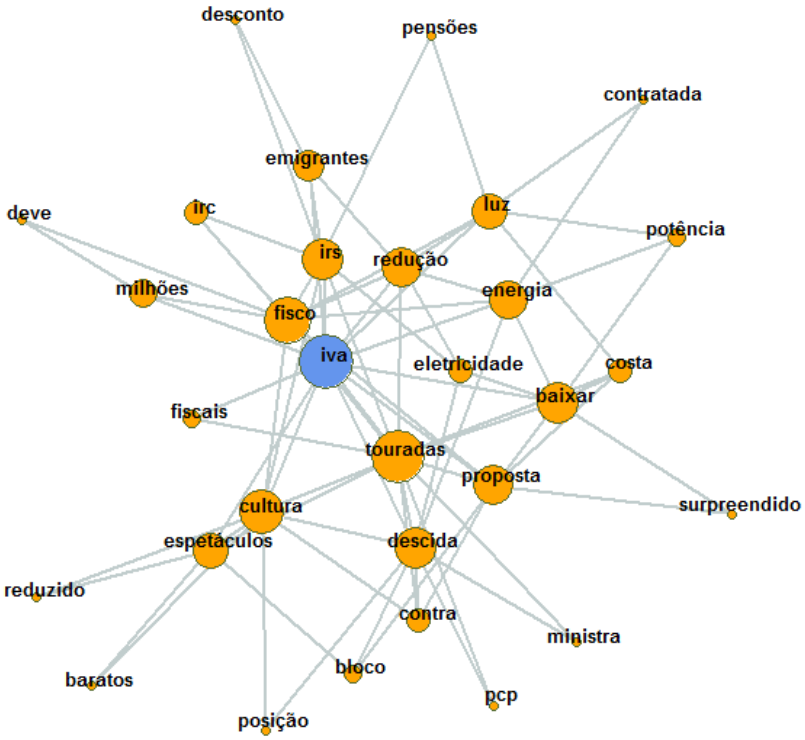


Figura 3.4 Grafo de coocorrências para o termo "IVA"

O termo "iva", como se pode ver na Figura 3.4, apresenta maiores índices de coocorrência com as palavras "touradas", "cultura", "redução" e "fisco", traduzindo o vasto conjunto de *tweets* publicados relacionados com a temática da redução do IVA aplicável em ingressos de espetáculos tauromáquicos.

Realizámos o mesmo procedimento para os termos "irs" e "fisco". Relativamente ao termo "irs", este apresenta maiores índices de coocorrência com as palavras "tabelas", "retenção", "2019", "salários", "pensões", "emigrantes" e "desconto", refletindo as preocupações manifestadas pelos utilizadores desta rede social em relação às retenções de IRS aplicáveis em 2019, mas também em relação ao novo regime de IRS aplicável a ex-emigrantes (artigo 12º-A do Código do IRS). Já o termo "fisco" apresenta maiores índices de coocorrência com as palavras relacionadas com dívidas ("deve", "dívida", "dívidas", "milhões"), excluindo as restantes *hashtags* utilizadas na pesquisa.

3.2 Implementação dos modelos selecionados

A implementação dos vários modelos testados foi precedida pela aplicação de algumas tarefas de pré-processamento, ao conjunto dos *tweets* que compõem a amostra que se descreve de seguida.

Na medida em que os objetivos fixados para este estudo consistem na categorização de *tweets* que expressam opiniões relacionadas com o funcionamento geral das autoridades fiscais portuguesas, limitar-nos-emos a analisar os resultados dos modelos eleitos: dois métodos de *supervised machine learning* (*Naive Bayes* e Support Vector Machines) e um método de baseado em léxicos em que será utilizada a biblioteca "LexiconPT" do software R.

3.2.1 Extração da amostra e classificação manual de *tweets*

A classificação de documentos não é um processo simples especialmente no caso dos *tweets* em que as mensagens são expressas através de textos muito curtos, com inúmeros erros ortográficos e em que muito poucos casos expressam uma opinião de forma clara. A fase inicial da classificação passa pela identificação e separação de textos do tipo informativo versus opinativo. Estes últimos são alvo de tratamento posterior quanto à sua objetividade, procurando identificar-se algumas componentes ou características que favoreçam o desempenho dos algoritmos de classificação

O processo de classificação manual dos *tweets* incidiu sobre uma amostra aleatória de 1.015 casos gerada a partir do conjunto de dados criado na fase de pré-processamento. Exportada a amostra para um ficheiro Excel nele realizámos a classificação manual dos *tweets* distinguindo aqueles que são relacionados com o tema em análise e simultaneamente manifestam uma opinião positiva (1) e negativa (-1), e os restantes que, por não respeitarem o tema ou apenas descreverem um facto ou informação, são classificados com uma pontuação igual a 0. Tendo em conta a diversidade e o grau de subjetividade observada na expressão das opiniões, optou-se por considerar como comentário de teor positivo, todo aquele que expressasse uma crítica negativa acerca de um comportamento negativo relacionado com o tema (ex.: comentário negativo acerca de um indivíduo que praticou fraude fiscal).

Dos 1.015 registos que continha a amostra, apenas 353 expressavam uma opinião relacionada com o tema em análise, e destes, apenas 125 manifestavam opinião positiva

(ou não negativa) representando cerca de 12% do total da amostra. Aos restantes 662 casos foi atribuída uma classificação neutra (class = 0).

3.2.2 Classificação com Naive *Bayes*

Neste trabalho, aplicámos o algoritmo *Naive Bayes* a uma amostra aleatória de 1.015 casos, gerada a partir do *dataset* criado na fase de pré-processamento e manualmente classificada (ver Anexo 4). Conforme referido na subsecção anterior, dos 1.015 registos que continha a amostra, apenas 353 expressavam uma opinião relacionada com o tema em análise, e destes, apenas 125 manifestavam opinião positiva (ou não negativa) representando cerca de 12% do total da amostra, sendo que a classe maioritária (tweets do tipo informativo ou não relacionados com o tema em análise) representam 65,2% do total da amostra.

O mesmo conjunto de dados foi transformado num objeto do tipo "Document Term Matrix" restringido aos termos com duas ou mais ocorrências ("freq <- findFreqTerms(dtm.train, 2)") e posteriormente repartido num conjunto de treino correspondente a 67% dos registos que compõem a amostra, e os restantes 33% dos registos foram incluídos no conjunto de teste. Recorrendo à função "naive*Bayes*" da biblioteca "R" "e1071", gerámos um classificador que permitiu prever a classificação dos *tweets* na amostra de teste.

Na aplicação da função da qual resulta o objeto do tipo "Document Term Matrix" atrás referido ponderámos os valores de acordo com a frequência dos termos. Nos trabalhos exploratórios testámos a aplicação de outros "pesos" da matriz documento/termo, com aplicação a medida "*weightTfIdf*" que pondera uma matriz de termo-documento com os valores TF-IDF correspondentes. No entanto, os resultados obtidos com utilização desta última medida apresentavam indicadores estatísticos de pior qualidade

Na aplicação do modelo utilizámos o suavizador de Laplace, de modo a contornar o problema da probabilidade condicionada quando os novos dados incluem valores de recurso que nunca ocorrem para um ou mais níveis de uma classe de resposta. O suavizador de Laplace adiciona um pequeno número a cada uma das contagens nas frequências de cada recurso, o que garante que cada recurso tenha uma probabilidade diferente de zero para cada classe. Normalmente, um valor de um ou dois para o suavizador de Laplace é suficiente.

Verifica-se que o significativo grau de assertividade total do modelo (78,87%), que designaremos por "N*Bayes* FTerms2", só é alcançado à custa da previsão efetuada para *tweets* não relacionados com o tema ou sem opinião (class = 0) conforme atesta o indicador de sensitividade. Este facto é também confirmado pela elevada percentagem do indicador "No Information Rate" na escolha da classe maioritária. A estatística "Kappa", igual a 0.0245, indica uma fraca precisão do classificador utilizado em função da sua precisão esperada. Em todo o caso, o grau de assertividade total do modelo (78,87%) é superior à proporção de elementos incluídos na classe maioritária que representam 65,2% do total da amostra.

A Estatística Kappa é uma medida de concordância usada em escalas nominais que nos indica quanto as observações se afastam das esperadas, fruto do acaso. Os valores da estatística Kappa variam de 0 a 1, sendo que "0" representa não haver concordância além do puro acaso, e "1" representa a concordância perfeita.

Tabela 3.1 Estatística Kappa

Valor de kappa	**Concordância**
0	Pobre
0 – 0,20	Ligeira
0,21 – 0,40	Considerável
0,41 – 0,60	Moderada
0,61 – 0,80	Substancial
0,81 – 1	Excelente

Fonte: Adaptado de Landis & Koch (2012)

Aplicando uma restrição ao número de termos com três ou mais ocorrências ("freq <- findFreqTerms(dtm.train, 3)") e mantendo o conjunto de treino correspondente a 66% da amostra, reduzimos ligeiramente a assertividade do modelo (74,5%), que designaremos por "N*Bayes* FTerms3", mas aumentamos a sensitividade e especificidade, quer para valores positivos quer negativos. Observa-se também que também um decréscimo no valor do indicador "No Information Rate", revelador de uma ligeira melhoria, confirmada pelo acréscimo registado na estatística "Kappa". De salientar também a melhoria da assertividade individual em todas as classes. A eficiência (*balanced accuracy*) média dos três níveis de classificação (positivo, negativo ou neutro) é de 53,4%.

Na Figura 3.5 apresenta-se o gráfico de distribuição das classes, comparando a classificação manual (real) com a prevista no modelo *Naive Bayes* (Previsão).

Figura 3.5 Gráfico de distribuição da classificação real e prevista pelo modelo Naive Bayes

O gráfico anterior apresenta um elevado grau de assertividade para a classe neutra (0), o que não se verifica nos restantes casos. De facto, os resultados para a classe "-1" evidenciam fraca precisão, apesar de não preverem casos classificados positivamente ("1"). Os resultados obtidos na classe "1" ainda são menos animadores uma vez que são classificados casos negativamente ("-1") em igual quantidade à dos corretamente classificados, para além da elevada classificação de casos como neutros.

Nas figuras seguintes apresentamos os histogramas das distribuições das três classes utilizadas no modelo.

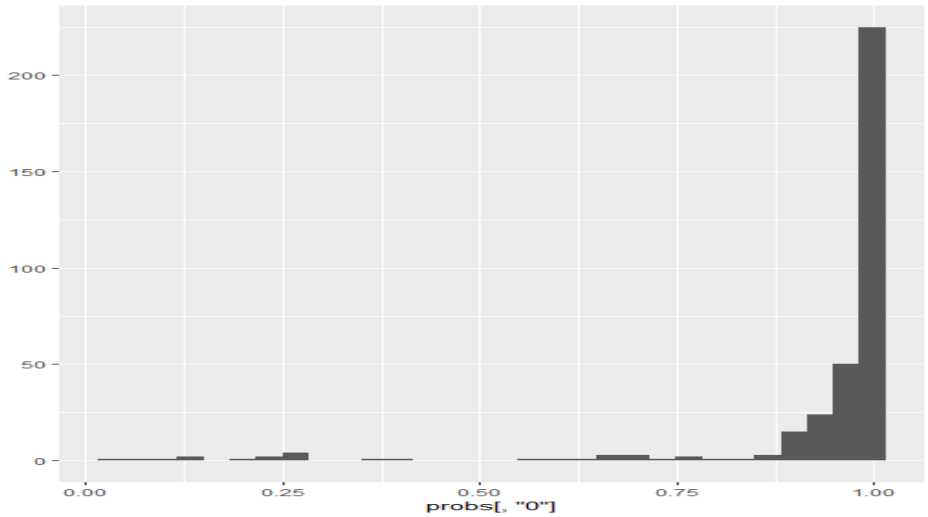


Figura 3.6 Histograma de distribuição da classe 0 (class = 0) do modelo

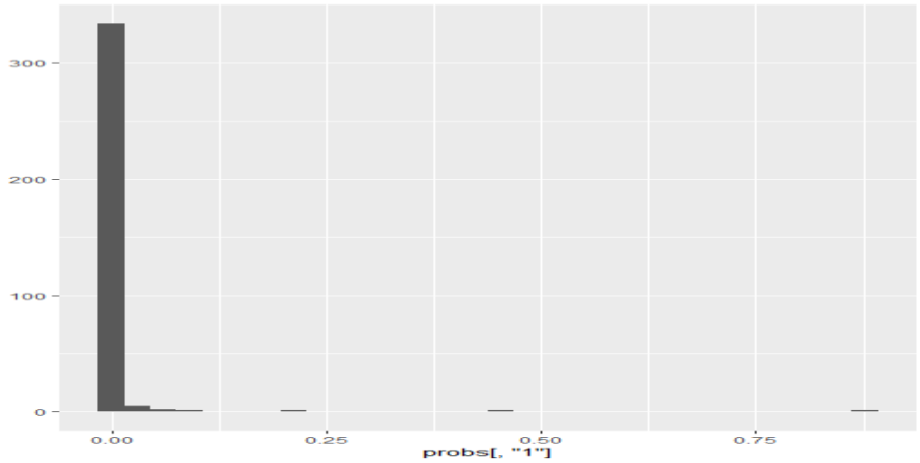


Figura 3.7 Histograma de distribuição da classe 1 (class = 1) do modelo

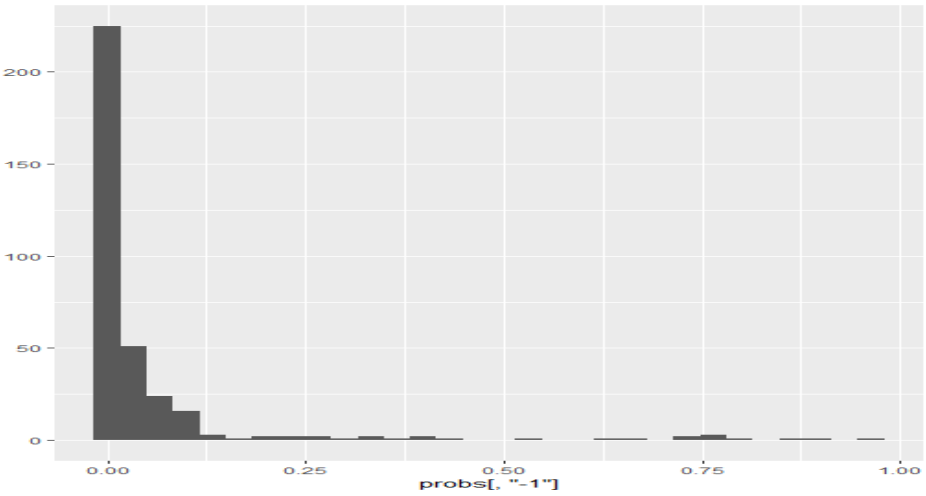


Figura 3.8 Histograma de distribuição da classe 1 (class = 1) do modelo

Os *tweets* com classificação neutra (*class* = 0) apresentam uma distribuição assimétrica à esquerda, com grande concentração da probabilidade junto do valor 1, revelador da elevada frequência dos *tweets* com esta classificação prevista. A classificação de *tweets* com opiniões negativas (*class* = -1) regista um comportamento inverso, com uma distribuição assimétrica à direita e elevada concentração da probabilidade junto do valor 0, apresentado, contudo, alguns casos com probabilidade que pode assumir o valor 1 (superior a 0,5). Já os *tweets* com opiniões positivas (*class* = 1) concentram a grande maioria das suas frequências numa probabilidade igual a zero, indiciador da (quase) inexistência de *tweets* com classificação positiva prevista.

3.2.3 Classificação com Lexicon-PT

O "Lexicon-PT" é atualmente uma biblioteca disponível no CRAN e dispõe de dois léxicos polarizados: oplexicon-PT (Souza & Vieira, 2012) e sentiLex-PT (Carvalho &

Silva, 2015). Esta técnica de classificação é baseada em recursos léxicos já disponíveis para utilização e são compostos por dicionários de palavras e/ou expressões que possuem classes pré-definidas e valores. Na prática, o algoritmo percorre os *tokens* que compõem o *corpus* e atribui-lhes o "valor sentimental" definido no léxico.

Na aplicação dos léxicos (ver Anexo 5) incluídos na biblioteca "Lexicon-PT" à amostra aleatória de 1.015 casos (228 *tweets* com opinião negativa e 125 casos com opinião positiva e 662 com classificação neutra) efetuámos uma simulação com atribuição de polaridade apenas em palavras que constassem dos 2 léxicos (oplexicon e sentiLex-PT) em simultâneo e um outro teste em que atribuímos a polaridade aos termos que constassem num dos 2 léxicos.

Na execução do primeiro teste apenas foram classificados 267 *tweets*, permanecendo 748 por classificar neste modelo que designámos por "LexPT Oplex". Pela análise tabular dos *tweets* classificados, verificámos que o modelo atribuiu pontuações inteiras variáveis entre -3 e 2. Tendo em vista a harmonização da escala comparativa entre modelos, optámos por estabelecer uma regra do tipo:

- Se (Classificação_Inicial > 1), então Classificação_Final =1
- Se (Classificação_Inicial < -1), então Classificação_Final =-1
- Noutros casos, então Classificação_Final = 0

Designámos de "Lexicon-PT Oplexicon" o modelo gerado com base na classificação atribuída pelo léxico "oplexicon", e o modelo gerado com base na classificação atribuída pelo léxico "sentiLex-PT" foi designádo de "LexPT sentLex".

Os resultados obtidos nos dois últimos testes são inferiores, em toda a linha, aos obtidos com o algoritmo *Naive Bayes*, já que classificaram apenas 267 dos 1.015 casos da amostra (26,3%).

Para a realização de um terceiro teste recorremos à biblioteca "Lexicon-PT" e utilizámos a mesma amostra, com polaridade atribuída por qualquer um dos léxicos utilizados, aos termos presentes na amostra. A estratégia consistiu na criação de duas tabelas auxiliares, com instruções do tipo "*inner join*", ligando os termos que constam da amostra aos termos presentes em cada um dos léxicos utilizados. A junção das duas tabelas gerou alguns elementos sem pontuação atribuída ("NA") numa das variáveis e que foram substituídos pela classificação atribuída pelo outro léxico. Também neste caso se verificou que o

modelo atribuiu pontuações inteiras variáveis entre -3 e 2, tendo-se aplicado a regra de classificação igual à utilizada no modelo anterior.

A pesquisa originou um *dataset* constituído por 880 *tweets* classificados neste modelo, que designámos de "LexPT SentOp", permanecendo 135 por classificar, sendo que: 52 deles foram classificados positivamente, igual quantidade foi classificada de forma negativa e 776 foram classificados como neutros (sent = 0). Nota-se, aqui, uma ligeira melhoria dos resultados em relação ao teste anterior. Contudo, os resultados obtidos permanecem inferiores aos obtidos com o algoritmo *Naive Bayes*. O número de casos classificados neste modelo aumentou significativamente para 86,7% do total da amostra.

3.2.4 Classificação com LexiNB

Experimentámos também a utilização do algoritmo *Naive Bayes*, aplicado ao *dataset* gerado na etapa anterior (880 *tweets*). O método utilizado (ver Anexo 6) consistiu na transformação desse conjunto de dados num objecto do tipo "Document Term Matrix" restringido aos termos com 3 ou mais ocorrências ("freq <- findFreqTerms(dtm.train, 3)") e posteriormente repartido num conjunto de treino correspondente a 66% dos registos e os restantes 34% foram incluídos no conjunto de teste. A polaridade prevista é igual à obtida nos 2 modelos (*Naive Bayes* e Lexicon-PT), quando a classificação é igual em ambos e assume o valor "0" nos restantes casos. A este modelo atribuímos a designação de "LexiNB".

Os resultados obtidos registam melhorias significativas ao nível do coeficiente de concordância (Kappa) entre os valores observados e os esperados, bem como da eficiência (*balanced accuracy*), quando comparada com a utilização isolada do *Naive Bayes* ou do dicionário "Lexicon-PT" sobre o conjunto de dados original.

Na Figura 3.9 apresentamos o gráfico de distribuição das classes, comparando a classificação manual (Real) com a prevista no modelo LexiNB (Previsão).

Figura 3.9 Gráfico de distribuição da classificação real e prevista pelo modelo LexiNB

Também neste gráfico se observa um elevado grau de assertividade para a classe neutra (0) mas com alguns casos mal classificados ("-1"). A classe "-1" apresenta um número de casos corretamente classificados equivalente aos que apresentam classificação incorreta. Repare-se ainda, que existe apenas 1 caso previstos com classificação positiva.

3.2.5 Classificação com SVM

À semelhança do descrito na subsecção anterior, aplicámos o classificador SVM à mesma amostra aleatória de 1.015 casos manualmente classificados (ver Anexo 7). Também aqui, transformámos o referido conjunto de dados num objeto do tipo "Document Term Matrix". Os testes efetuados evidenciaram melhores resultados aplicando a restrição de termos frequentes igual a 2 ("freq <- findFreqTerms(dtm.train, 2)"). Testámos também a aplicação da medida TF-IDF na matriz de termo-documento mas os resultados obtidos apresentavam indicadores estatísticos de pior qualidade.

Aquele objeto foi posteriormente repartido num conjunto de treino e teste em iguais proporções às utilizadas na subsecção anterior. Recorrendo à função "svm" da biblioteca "R" "e1071", e após vários testes para otimização dos parâmetros correspondentes, gerámos classificadores com Kernel linear, gaussiano (Radial Basis) e polinomial posteriormente aplicados ao conjunto de teste para avaliação dos resultados. Note-se que os valores da matriz documento/termo não foram normalizados em nenhum dos modelos uma vez que os testes iniciais apresentavam piores resultados com valores normalizados.

3.2.5.1 SVM – Kernel linear

A função-regra deste modelo apresenta a seguinte forma:

K_LIN = svm(cat~., data=train, method="C-classification", *kernel = "linear", cost = 1, scale = FALSE)*

cost representa a constante do termo de regularização na fórmula de Lagrange e representa o custo de violação de restrições e o seu valor padrão é igual a 1.

Apesar do modelo ter sido construído sem quaisquer restrições ou especificações adicionais, regista-se um significativo grau de assertividade total do modelo (75,58%) que designaremos por "K_LIN", e o valor da estatística Kappa situa-se num nível considerável (29,32%). O indicador de eficiência (*balanced accuracy*) supera em mais de 17% o do melhor modelo testado com o algoritmo *Naive Bayes*, situando-se em 62,6%.

Na Figura 3.10 apresentamos o gráfico de distribuição das classes, comparando a classificação manual (real) com a previsto no modelo SVM linear (previsão).

Figura 3.10 Gráfico de distribuição da classificação real e prevista pelo modelo SVM Linear

Também neste gráfico se observa um elevado grau de assertividade para a classe neutra (0) persistindo a ocorrência de alguns casos mal classificados ("-1" ou "1"). Apesar de terem melhorado em relação ao modelo *Naive Bayes*, os resultados para a classe "-1" possuem ainda fraca precisão. Também os resultados para a classe "1" melhoraram ligeiramente. No entanto, continuam a ser classificados casos negativamente ("-1") em quantidade sensivelmente igual à dos corretamente classificados, para além da elevada classificação de casos como neutros.

Para determinação do melhor modelo linear (best_LIN) utilizámos um algoritmo que percorre uma lista para o coeficiente "cost" = {0.001, 0.01, 0.1, 1, 5, 10, 20, 100} que fixou o valor 0.001 para aquele coeficiente. Apesar da assertividade total ter melhorado ligeiramente (76,74%), tal só acontece devido ao acréscimo na assertividade para os *tweets* com "carga" neutra uma vez que o rácio de deteção de *tweets* não neutros é igual a zero neste modelo.

3.2.5.2 SVM – Kernel gaussiano (Radial Basis)

A estrutura da função-tipo para aplicação deste método é muito similar à função "svm" com Kernel linear. A função utilizada para determinação do modelo designado K_RAD tem a seguinte estrutura:

```
K_RAD <- svm(cat~., data=train, method="C-classification", kernal="radial", gamma=0.1, cost=10, scale=FALSE)
```

Gamma é um parâmetro necessário para todos os Kernels, exceto linear, cujo padrão é igual a 1/(dimensão de dados).

Também neste caso, utilizámos uma lista para o coeficiente "*cost*" = {0.001, 0.01, 0.1, 1, 5, 10, 20, 100} e outra lista para o parâmetro "*gamma*" = {0.1,0.5,1,2,3,4}, para determinação do melhor modelo gaussiano (best_RAD). Curiosamente, observamos o mesmo comportamento registado com a utilização de kernel linear, com ligeiro acréscimo da assertividade total à custa do aumento registado no rácio de deteção de *tweets* não neutros. Os valores observados para a estatística Kappa e eficiência são bastante inferiores ao modelo "svm" com Kernel linear.

3.2.5.3 SVM – Kernel polinomial

A estrutura da função-tipo para aplicação deste método é também idêntica à função "svm" com Kernel linear. A função utilizada para determinação deste modelo, designado "K_POL", tem a seguinte estrutura:

```
K_POL <- svm(cat~., data=sms_train1, type='C-classification', kernel = 'polynomial', degree=8, gamma=0.1, coef0=1, scale=FALSE)
```

Também aqui, utilizámos uma lista para o coeficiente "*cost*" = {0.001, 0.01, 0.1, 1, 5, 10, 20, 100} e outra lista para o parâmetro "*gamma*" = {0.1,0.5,1,2,3,4}, para determinação do melhor modelo polinomial (best_POL) e observamos o mesmo comportamento

registado com a utilização de Kernel linear e radial/gaussiano, com ligeiro acréscimo da assertividade total à custa do aumento registado no rácio de deteção de *tweets* não neutros. Apesar da assertividade total deste modelo ser superior à do modelo linear "K_LIN", este último modelo apresenta valores superiores para a estatística Kappa e na eficiência (*balanced accuracy*).

3.2.6 Análise comparativa dos resultados obtidos

Este capítulo destina-se à apresentação dos resultados obtidos em cada um dos modelos testados e que foram objeto de descrição detalhada no capítulo anterior.

3.2.6.1 Resultados para os modelos Naive Bayes e Lexicon-PT

No quadro seguinte apresentamos um resumo dos principais indicadores estatísticos obtidos na aplicação dos 6 modelos testados:

Tabela 3.2 Resumo comparativo dos resultados

Indicador	**N*Bayes* FTerms2**	**N*Bayes* FTerms3**	**LexPT Oplex**	**LexPT sentLex**	**LexPT SentOp**	**LexiNB**
Accuracy	0,7420	0,7449	0,6479	0,6367	0,6841	0,7550
No Information Rate	0,7391	0,7391	0,6966	0,6966	0,7625	0,7483
Kappa	0,0236	0,1039	0,0160	-0,0077	-0,0074	0,1526
Sensitivity						
Class: -1	0,0179	0,0893	0,1053	0,0877	0,0638	0,1569
Class: 0	1,0000	0,9804	0,8978	0,8871	0,8763	0,9686
Class: 1	0,0000	0,0588	0,0000	0,0000	0,0735	0,0417
Specificity						
Class: -1	0,9965	0,9758	0,9571	0,9571	0,9418	0,9636
Class: 0	0,0222	0,1000	0,0988	0,0864	0,1005	0,1600
Class: 1	1,0000	1,0000	0,9506	0,9424	0,9421	0,9964
Detection Rate						
Class: -1	0,0029	0,0145	0,0225	0,0187	0,0102	0,0269
Class: 0	0,7391	0,7246	0,6255	0,6180	0,6682	0,7248
Class: 1	0,0000	0,0058	0,0000	0,0000	0,0057	0,0034
Balanced Accuracy						
Class: -1	0,5072	0,5325	0,5312	0,5224	0,5028	0,5602
Class: 0	0,5111	0,5402	0,4983	0,4868	0,4884	0,5643
Class: 1	0,5000	0,5294	0,4753	0,4712	0,5078	0,5190

Os resultados obtidos com o modelo designado de "LexiNB" registam um aumento significativo (196%) da estatística Kappa, que mede a precisão do classificador utilizado em função da sua precisão esperada, situando-se agora num escalão moderado, e um incremento médio de cerca de 14% na sua eficiência (Balanced Accuracy = (sensibilidade + especificidade)/2), que resulta essencialmente do aumento na deteção de opiniões negativas, permanecendo, no entanto, com indicadores muito baixos. Observa-se também, uma ligeira melhoria na assertividade total, quando comparada com a utilização isolada do *Naive Bayes* ou do dicionário Lexicon-PT.

3.2.6.2 Resultados para os modelos SVM

No quadro seguinte apresentamos um resumo dos principais indicadores estatísticos obtidos na aplicação dos 6 modelos testados.

Tabela 3.3 Resumo comparativo dos resultados dos modelos SVM

Indicador	**K_LIN**	**best_LIN**	**K_RAD**	**best_RAD**	**K_POL**	**best_POL**
Accuracy	0.7558	0.7674	0.7703	0.7674	0.7762	0.7703
No Information Rate	0.7674	0.7674	0.7674	0.7674	0.7674	0.7674
Kappa	0.2932	0.0000	0.1028	0.0160	0.1092	0.0498
Sensitivity						
Class: -1	0.3019	0.0000	0.1132	0.0000	0.0566	0.0377
Class: 0	0.8977	1.0000	0.9773	0.9962	0.9886	0.9924
Class: 1	0.2593	0.0000	0.0370	0.0370	0.1111	0.0370
Specificity						
Class: -1	0.9107	1.0000	0.9794	0.9966	0.9897	0.9931
Class: 0	0.4250	0.0000	0.0875	0.0125	0.0875	0.3750
Class: 1	0.9622	1.0000	1.0000	1.0000	0.9968	1.0000
Detection Rate						
Class: -1	0.0465	0.0000	0.0174	0.0000	0.0087	0.0058
Class: 0	0.6890	0.7674	0.7500	0.7645	0.7587	0.7616
Class: 1	0.0204	0.0000	0.0029	0.0029	0.0087	0.0029
Balanced Accuracy						
Class: -1	0.6063	0.5000	0.5463	0.4983	0.5231	0.5154
Class: 0	0.6614	0.5000	0.5324	0.5044	0.5381	0.5150
Class: 1	0.6107	0.5000	0.5185	0.5185	0.5540	0.5185

O modelo SVM construído sem quaisquer restrições ou especificações adicionais (modelo-regra) designado por "K_LIN", regista um significativo grau de assertividade (75,58%) e classifica toda a amostra. O valor da estatística Kappa situa-se num nível considerável (29,32%), apesar de ligeiramente inferior à do modelo "LexiNB". Também indicador de eficiência (*balanced accuracy*) média para as 3 classificações (positiva, negativa e neutra) é ligeiramente superior no modelo "LexiNB".

CONCLUSÕES

O objetivo principal na realização deste trabalho consistia na aplicação de técnicas classificação de textos que possibilitassem o acompanhamento e monitorização de *posts* publicados na rede social Twitter acerca do funcionamento geral das autoridades fiscais portuguesas, recorrendo a dois métodos de *supervised machine learning* (*Naive Bayes* e *Support Vector Machines* (SVM)) e um método de baseado em léxicos, em que se utilizou a biblioteca "LexiconPT" do software R.

Este estudo permitiu concluir que os utilizadores da rede social Twitter abordam, com maior predominância, a temática dos impostos e da carga fiscal, sendo menos frequentes os comentários sobre o funcionamento da administração fiscal portuguesa, sobressaindo, nestes últimos, a preocupação dos utilizadores em relação aos elevados montantes das dívidas ao fisco.

Na nossa análise observámos que os resultados obtidos individualmente, quer pelo algoritmo *Naive Bayes* quer pela utilização da biblioteca "Lexicon-PT" não são suficientemente interessantes do ponto de vista estatístico. Realçamos, contudo, a melhoria significativa dos resultados obtidos com aplicação da abordagem bietápica proposta neste trabalho, designada de "LexiNB", em que aplicámos o algoritmo *Naive Bayes* ao conjunto de dados gerado com a utilização dos dois léxicos disponíveis na biblioteca "Lexicon-PT".

Da análise dos resultados suportados nas previsões obtidas com recurso à abordagem bietápica "LexiNB", concluímos que as opiniões negativas se manifestaram essencialmente acerca da redução da taxa de IVA nas touradas, enquanto as opiniões positivas se manifestaram através de críticas negativas acerca de comportamentos negativos relacionados com a fraude em IVA e com dívidas ao fisco.

A aplicação do classificador SVM apresenta algumas vantagens em relação ao modelo "LexiNB" por nós proposto, uma vez que permite classificar toda a amostra enquanto o modelo "LexiNB" apenas classifica cerca de 87% da amostra. No entanto, este último apresenta melhor assertividade total e maior precisão do classificador utilizado em função da sua precisão esperada (estatística Kappa).

Como principais contributos deste trabalho salientamos a validação das hipóteses formuladas inicialmente, nomeadamente a hipótese 1 sobre a possibilidade de classificar

os *tweets* relativos às autoridades fiscais portuguesas de acordo com o sentimento expresso nos mesmos, em positivos, negativos ou neutros e a hipótese 2 sobre a possibilidade do acompanhamento automático das opiniões manifestadas pelos utilizadores da rede social Twitter.

Destacamos ainda, uma nova abordagem bietápica proposta neste artigo, designada de "LexiNB, da qual resultou a escrita do nosso artigo intitulado "LexiNB - Uma abordagem bietápica de classificação de sentimentos em *tweets* relacionados com as autoridades fiscais portuguesas", submetido e aceite na 19.ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI'2019).

No decurso deste trabalho deparámo-nos com alguns constrangimentos e limitações que descrevemos de seguida.

A limitação do número de carateres posta pela rede social Twitter introduz alguns constrangimentos na análise *n-grams* impossibilitando as análises que envolvam a geração de *n-grams* com n superior a 4, uma vez que a sua quantidade é muito diminuta. Também as técnicas de "*POS-Tagging*" estão, de alguma forma, comprometidas no nosso trabalho derivado à grande desproporção entre a quantidade de "nomes" e de "verbos" ou "adjetivos". Por outro lado, a utilização de léxicos de opinião para a língua portuguesa, cujas palavras têm já as respetivas polaridades associadas, estão limitados ao tamanho da respetiva base de dados.

A elevada quantidade de erros ortográficos e a inexistência de corretores eficazes para a língua portuguesa, para além da pequena quantidade de *tweets* que expressam opinião de forma clara, são outras das limitações que encontramos no decurso deste trabalho.

Em trabalhos futuros, perspetivamos o refinamento dos termos da pesquisa inicial de *tweets*, um alargamento do tamanho da amostra manualmente classificada e a aplicação do modelo misto atrás descrito como forma de colmatar as insuficiências encontradas e de melhorar o desempenho global na classificação dos *tweets*. Também não está posta de parte a hipótese de utilização de algoritmos de aprendizagem não supervisionada que possam contribuir para a melhoria do desempenho do nosso modelo. Perspetivamos, também, a análise dos *tweets* englobando os denominados símbolos de emoção (*emoticons* e *emojis*).

Após a apresentação do presente relatório, tencionamos submeter uma proposta, junto da Ex.ma Srª Diretora-Geral dos Impostos, para implementação desta metodologia, eventualmente melhorada em função dos trabalhos futuros propostos, visando a evolução da organização (Autoridade Tributária e Aduaneira) no modelo de maturidade na utilização de redes sociais proposto pela OCDE.

REFERÊNCIAS BIBLIOGRÁFICAS

Álvarez, A. C. (2007). *Extração de informação de artigos científicos: um abordagem baseada em indução de regras de etiquetagem.* São Paulo, Brasil: Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo.

Arunachalam, R., & Sarkar, S. (2013). The New Eye of Government: Citizen Sentiment Analysis in Social Media. *IJCNLP 2013 Workshop on Natural Language Processing for Social Media (SocialNLP)* (pp. 23-28). Nagoya, Japan,: IJCNLP 2013 Workshop on Natural Language Processing for Social Media (SocialNLP).

Bing, L. (2012). *Sentiment Analysis and Opinion Mining,*. USA: Morgan & Claypool Publishers.

Carvalho, P., & Silva, M. J. (2015). Sentilex-PT: Principais características e potencialidades. *Linguística, Informática e Tradução: Mundos que se cruzam, Oslo Studies in Language 7(1),* pp. 425-438.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., & Wirth, R. (2000). *CRISP-DM 1.0: Step-by-step data mining guide.* EUA: CRISP-DM consortium.

Dellia, P., & Tjahyanto, A. (2017). Tax Complaints Classification on Twitter Using Text Mining. *IPTEK, Journal of Science, Vol. 2, No. 1*.

Feldman, R., Fresko, M., Hirsh, H., Aumann, Y., Liphstat, O., Schler, Y., & Rajman, M. (1998). Knowledge Management: A Text Mining Approach. *2nd Int. Conf. on Practical Aspects of Knowledge Management (PAKM98)* (pp. 1-10). Basel, Switzerland: U. Reimer.

Filho, J. A. (2014). *Mineração de textos: análise de sentimento utilizando tewwets referentes à copa do mundo 2014.* Qixada, Ceará, Brasil: Universidade Federal do Ceará.

Fortuny, E. J., Smedt, T. D., Martens, D., & Daelemans, W. (2012). Media coverage in times of political crisis: A text mining approach. *ELSEVIER - Expert Systems with Applications*, 11616–11622.

Fortuny, E. J., Smedt, T. D., Martens, D., & Daelemans, W. (2014). Evaluating and understanding text-based stock price prediction models. *ELSEVIER - Information Processing and Management*, 426-441.

Forum on Tax Administration. (2011). *Social Media Technologies and Tax Administration.* OECD - Centre for Tax Policy and Administration.

Gaizauskas, R., & Wilks, Y. (1998). Information Extraction: Beyond Document Retrieval. *Computational Linguistics Society of R.O.C.*, 17-60.

Gamallo, P., Garcia, M., & Fernandez-Lanza, S. (2013). *TASS: A Naive-Bayes strategy for sentiment analysis on Spanish tweets.* https://gramatica.usc.es/~gamallo/artigos-web/TASS2013.pdf.

Gharehchopogh, F. S., & Khalifelu, Z. A. (2011). Analysis and Evaluation of Unstructured Data: Text Mining versus Natural Language Processing. *International Journal of Academic Research in Computer Engineering*.

Gomes, H. J. (2012). *Text Mining: Análise de Sentimentos na classificação de notícias.* Lisboa: Instituto Superior de Estatística e Gestão de Informação da Universidade Nova de Lisboa.

Grishman, R. (1998). Information Extraction: Techniques and Challenges. *Computer Science Department - New York University*, 1-18.

Hu, M., & Liu, B. (2004). Mining and Summarizing Customer Reviews. *Department of Computer Science - University of Illinois at Chicago*.

Indurkhya, N., & Damerau, F. J. (2010). *Handbook of Natural Language Processing - Second Edition.* USA: Chapman & Hall/CRC.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Universitat Dortmund - Informatic LS8*.

Khan, W., Daud, A., Nasir, J. A., & Amjad, T. (2016). A survey on the state-of-the-art machine learning models in the context of NLP. *Kuwait J. Sci. 43*, 95-113.

Khasawneh, R. T., & Abu-Shanab, E. A. (2013). E-Government and Social Media Sites: The Role and Impact. *World Journal of Computer Application and Technology*, 10-17.

Kim, G.-H., Trimi, S., & Chung, J.-H. (2014). Big-Data applications in the government sector. *Communications of the acm | march 2014 | vol. 57 | no.3*, 78-85.

Kivinen, J., Warmuth, M., & Auer, P. (1997). The perceptron algorithm vs. Winnow: linear vs. logarithmic mistake bounds when few input variables are relevant. *NeuroCOLT Technical Report Series - Department of Computer Science - University of London*.

Kushmerick, N., & Thomas, B. (2003). Adaptive information extraction: Core technologies for information agents. *Lecture Notes in Computer Science - 2586*, 79-103.

Landis, J. R., & Koch, G. G. (2012). The Measurement of Observer Agreement for Categorical Data. *JSTOR: http://www.jstor.org/stable/2529310* , 159-174.

Liddy, E. D. (2001). Natural Language Processing. Em E. D. Liddy, *Natural Language Processing.* NY: Ed. Marcel Decker, Inc.

Lima, V. G. (2013). *Ubibusanalysis - Uma ferramenta de interpretação de mensagens de trânsito com Análise de Sentimentos.* Recife, Brasil: Universidade Federal de Pernambuco.

Liu, B. (2010). Opinion Mining and Sentiment Analysis: NLP Meets Social Sciences. *Department of Computer Science University Of Illinois at Chicago*.

Liu, B. (2011). Sentiment analýsis and opinion mining. *Department of Computer Science - University Of Illinois at Chicago*.

Liu, B., & Zhang, L. (2012). A Survey of Opinion Mining and Sentiment Analysis. Em B. Liu, & L. Zhang, *Mining Text Data.* USA.

Loper, E., & Bird, S. (2002). NLTK: The Natural Language Toolkit. Em *NLTK: The Natural Language Toolkit* (pp. 63-70). https://www.semanticscholar.org/paper/NLTK%3A-The-Natural-Language-Toolkit-Loper-Bird/: Semantic Scholar.

Lorena, A. C. (2006). *Investigação de estratégias para a geração de máquinas de vetores de suporte multiclasses.* São Paulo, Brasil: Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo.

Mitchell, T. M. (1997). *Machine Learning.* NY, USA: McGraw-Hill Science/Engineering/Math.

Nilsson, N. J. (1998). *Introduction to machine learning - Draft of a proposed textbook.* Stanford, CA, USA.

Pan, L. (2012). *Sentiment Analysis in Chinese.* Waltham, Massachusetts, USA: Faculty of the Graduate School of Arts and Sciences - Brandeis University - Department of Computer Science.

Pang, B. (2006). *AutomaticC Analysis of Document Sentiment.* USA: Faculty of the Graduate School of Cornell University.

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? Sentiment Classi¯cation using Machine Learning Techniques. *Conference on Empirical Methods in Natural Language Processing* (pp. 79-86). Philadelphia, USA: Association for Computational Linguistics.

Parikh, R., & Movassate, M. (2009). *Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques.* https://nlp.stanford.edu/courses/cs224n/2009/fp/19.pdf.

Patheja, P., Waoo, A., & Garg, R. (2012). Analysis of part of speech tagging. *International Conference on Intuitive Systems & Solutions (ICISS).* Mumbai, INDIA: International Journal of Computer Applications® (IJCA).

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the Poor Assumptions of Naive *Bayes* Text Classi. *Twentieth International Conference on Machine Learning.* Washington DC: Artificial Intelligence Laboratory - MIT - Cambridge.

Russell, M. A. (2013). *Mining the Social Web - 2ª Edition.* USA: O'Reilly.

Russell, S. J., & Norvig, P. (2010). *Artificial Intelligence: A Modern Approach - Third Edition.* Prentice Hall.

Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, pp. 513-523.

Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys, Vol. 34, No. 1*, 1-47.

Souza, M., & Vieira, R. (2012). *Sentiment Analysis on Twitter Data for Portuguese Language*. https://www.researchgate.net/publication/262175717.

Sulova, S., Todoranova, L., Penchev, B., & Nacheva, R. (2017). Using text mining to classify research papers. *17th International Multidisciplinary Scientific GeoConference SGEM 2017.* Varna, Bulgaria: University of Economics - Varna, Bulgaria.

Talbot, R., Acheampong, C., & Wicentowski, R. (2015). SWASH: A Naive *Bayes* Classifier for Tweet Sentiment Identification. *9th International Workshop on Semantic Evaluation (SemEval 2015),* (pp. 626-630). Denver, Colorado: Association for Computational Linguistics.

Troussas, C., Virvou, M., Espinosa, K. J., Llaguno, K., & Caro, J. (2013). Sentiment analysis of Facebook statuses using Naive *Bayes* classifier for language learning. *IISA 2013.* Piraeus, Greece: IISA 2013.

Vapnik, V. N. (1998). *Statistical Learning Theory.* NY: J. Wiley.

Vapnik, V. N. (2010). *Yhe Nature of Statistical Learning Theory.* NY: Springer.

Varela, P. d. (2012). *Sentiment Analysis.* Lisboa: Instituto Superior Técnico.

ANEXOS

ANEXO 1 - CÓDIGO "R" PARA CRIAÇÃO DO DATASET FINAL; PRÉ-TRATAMENTO DOS TWEETS

```
setwd("C:/…/5_TESE/AnaliseSentimentos")

library("readxl")
library(splus2R)
library(stringr)
library(DescTools)
library(data.table)
library(dplyr)
library(tm)
library(plyr)
library(sqldf)
library(tidytext)
library(tidyr)
library(ptstem)
library(udpipe)
library("textcat")

twt <- read_excel("teste20190301.xlsx", sheet="teste20190301") # Ver página 23
twt1 <- twt
nrow(twt1)
# [1] 84664

twt1$screenName <- lowerCase(twt1$screenName)
twt1$text <- lowerCase(twt1$text)
twt1$text <- str_squish(twt1$text)
twt1$text <- StrTrim(twt1$text, pattern = " \t\n", method = "both")
twt2 <- twt1[-grep("(iva|irs|irc|fisco|tributarias|fiscais|finanças)", twt1$screenName), ]
twt3 <- twt2[grep("(iva|irs|irc|fisco|tributarias|fiscais|finanças|financas)", twt2$text), ]
twt4 <- twt3[-
grep("(brasil|espanha|angola|moçambiq|eua|alckmin|ciro|bolsonaro|haddad|estadual)",
twt3$text), ]
twt4$text <- gsub("\\https.*","",twt4$text)
twt4$text <- gsub("<.*?>","",twt4$text)
twt4$text <- gsub("rt @.*?:","",twt4$text)
twt4$text <- gsub("@.*? ","",twt4$text)
twt4$text <- StrTrim(twt4$text, pattern = " \t\n", method = "both")
twt5 <- twt4[grep("(iva|irs|irc|fisco|tributarias|fiscais|finanças)", twt4$text), ]
```

```
twitter_df <- twt5[-
grep("(renegade|artisticvoiceinstitute|hybrid|institutodavozartís|fiscocontabeis|espanhol|quenian
o|grécia|spb|chuchu|cizânia)",twt5$text), ]
twitter_df$text = gsub("q ", "que ", twitter_df$text)
twitter_df$text = gsub("#", "", twitter_df$text)
twitter_df$text = gsub("pra ", "para ", twitter_df$text)
twitter_df$text = gsub("electricidade ", " eletricidade ", twitter_df$text)
twitter_df$text <- gsub("[^[:alnum:][:blank:]?&/\\-]", " ", twitter_df$text) # remover não-UTF8
twitter_df$text <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", twitter_df$text, perl=TRUE)
dt_proc <- subset(twitter_df, nchar(text)>=40)
dtx <- as.data.table(dt_proc)
dt_x <- dtx[, paste(collapse=""), by = c("doc_id", "text")]
dt1 <- dt_x %>% select(text)
dt2 <- as.data.table(dt1)
dt3 <- dt2[, paste(collapse=""), by = text]
train <- dt3
train <- mutate(train, id = rownames(train))
stpw <- stopwords(kind = "pt")
docs = tm::Corpus(tm::VectorSource(train$text))
docs <- tm_map(docs, removeWords, stpw)
docs <- tm_map(docs, removeNumbers)
docs <- tm_map(docs, removePunctuation)
docs <- tm_map(docs, stripWhitespace)


docs2 <- data.frame(matrix(unlist(docs), byrow=T))
docs2 <- mutate(docs2, id = rownames(docs2))
colnames(docs2) <- c("texto", "id")
docs3 <- docs2[1:12663,]
dff <- sqldf("SELECT train.text, train.id, docs3.texto FROM train INNER JOIN docs3 WHERE
train.id = docs3.id")
nrow(dff)
# [1] 12663
```

 # Pré-processamento: frases excluídas que incluem certas palavras recorrendo à análise n-gram:

 # Aplicação dos resultados da análise de Pré-processamento (n-grams = 2)

```
docs22 = tm::Corpus(tm::VectorSource(dff$texto))
docs_2 <- data.frame(matrix(unlist(docs22), byrow=T))
docs_2 <- mutate(docs_2, id = rownames(docs_2))
```

```
colnames(docs_2) <- c("texto2", "id")

docs_3 <- docs_2[1:12663,]

df_f <- sqldf("SELECT dff.text, dff.id, docs_3.texto2 as texto FROM dff INNER JOIN docs_3 WHERE dff.id = docs_3.id")

doisgram <- df_f %>% select(texto) %>% unnest_tokens(bigram, texto, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)

doisgram2  <- doisgram %>% separate(bigram, c("word1", "word2"), sep = " ")

print(tbl_df(doisgram2), n=300)

doisgram22 <- doisgram2[c(8,66,70,90,104,120,159,169,193,200,201,204,234,263,270), ]

teste_dg <- sqldf("SELECT word1, word2 from doisgram22")

teste2_dg <- sqldf("SELECT df_f.text, df_f.id, df_f.texto FROM df_f INNER JOIN teste_dg WHERE df_f.texto LIKE '%' ||teste_dg.word1||'%' AND df_f.texto LIKE '%' ||teste_dg.word2||'%' ")

print(tbl_df(teste2_dg), n=50)

zz2 <- anti_join(df_f, teste2_dg, by="texto")

nrow(zz2)

# [1] 12161
```

 # Aplicação dos resultados da análise de Pré-processamento (n-grams = 3)

```
tresgram <- zz2 %>% select(texto) %>% unnest_tokens(bigram, texto, token = "ngrams", n = 3) %>% dplyr::count(bigram, sort = TRUE)

tresgram2  <- tresgram %>% separate(bigram, c("word1", "word2", "word3"), sep = " ")

print(tbl_df(tresgram2), n=300)

trigram2 <- tresgram2[c(76,77,88,92,102,103,151,176,183), ]

teste_tg <- sqldf("SELECT word1, word2, word3 from trigram2")

teste2_tg <- sqldf("SELECT zz2.text, zz2.id, zz2.texto FROM zz2 INNER JOIN teste_tg WHERE zz2.text LIKE '%' ||teste_tg.word1||'%' AND zz2.text LIKE '%' ||teste_tg.word2||'%' AND zz2.text LIKE '%' ||teste_tg.word3||'%'")

print(tbl_df(teste2_tg), n=50)

zz3 <- anti_join(zz2, teste2_tg, by="texto")

nrow(zz3)

# [1] 12103
```

 # Aplicação dos resultados da análise de Pré-processamento (tf_idf)

```
ddgram <- zz3 %>% select(texto) %>% unnest_tokens(bigram, texto, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)

ddgram2 <- ddgram %>% separate(bigram, c("word1", "word2"), sep = " ")


ddgram3 <- ddgram2 %>% bind_tf_idf(word1, word2, n) %>% arrange(desc(tf_idf))

bigramx <- as.data.frame(ddgram3)
```

```
head(bigramx, 100)

bigramx2 <-
bigramx[c(4,7:14,18:23,25:27,35:38,48,52:58,63,66,71:75,78,86,87,90:94,96,98,99), ]

teste_tf <- sqldf("SELECT word1, word2 from bigramx2")

teste2_tf <- sqldf("SELECT zz3.text, zz3.id, zz3.texto FROM zz3 INNER JOIN teste_tf WHERE
zz3.texto LIKE '%' ||teste_tf.word1||'%' AND zz3.texto LIKE '%' ||teste_tf.word2||'%' ")

print(tbl_df(teste2_tf), n=50)

zz4 <- anti_join(zz3, teste2_tf, by="texto")

zzz <- sqldf("SELECT zz4.text, zz4.id FROM zz4")

train1 <- zzz

colnames(train1) <- c("text", "doc_id")

train1 <- mutate(train1, id = rownames(train1))

docsx = tm::Corpus(tm::VectorSource(train1$text))

docs2x <- data.frame(matrix(unlist(docsx), byrow=T))

docs2x <- mutate(docs2x, id = rownames(docs2x))

colnames(docs2x) <- c("texto", "id")

nrow(docs2x)

# [1] 12017

docs3x <- docs2x[1:12016,]

dft <- sqldf("SELECT train1.text, train1.doc_id, train1.id, docs3x.texto FROM train1 INNER JOIN
docs3x WHERE train1.id = docs3x.id")

nrow(dft)

[1] 12016
```

#Pré-processamento: frases excluídas que não estão escritas na língua portuguesa recorrendo à análise *POS-Tagging*:

```
dft$texto <- gsub("[[:punct:]]", " ", dft$texto)

dft$texto <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", dft$texto, perl=TRUE)

udmodel_pt <- udpipe_load_model(file = "portuguese-bosque-ud-2.3-181115.udpipe")

x <- udpipe_annotate(udmodel_pt, dft$texto, doc_id = paste("doc", dft$doc_id, sep = ""))

y <- as.data.frame(x)

str(y)

y$lang <- textcat(y$sentence)

sim_pt <- subset(y, lang == "portuguese")

nao_pt <- subset(y, lang != "portuguese")

d_npt <- as.data.table(nao_pt)

dt_npt <- d_npt[, paste(collapse=""), by = c("sentence", "doc_id")]

dt_npt %>% count(lang, sort=T)

d_spt <- as.data.table(sim_pt)

d_spt$stempt <- ptstem_words(d_spt$token, algorithm = "porter", complete = T)
```

```
dt_spt <- d_spt[, paste(collapse=""), by = c("sentence", "doc_id")]
dt_final <- subset(dt_spt, nchar(sentence)>=38)
nrow(dt_final)
# [1] 11464

write.csv2(dt_final, "C:/,,,/5_TESE/AnaliseSentimentos/dt_final.csv")
```

ANEXO 2 - CÓDIGO "R" PARA ANÃLISE "N-GRAMS "

```
setwd("C:/…/5_TESE/AnaliseSentimentos")

library("readxl")
library(tm)
library(plyr)
library(tidytext)
library(tidyr)
library(sqldf)
library(data.table)
library(dplyr)
```

 # n-grams – análise genérica

```
twt1 <- read_excel("teste20190301.xlsx", sheet="teste20190301") # Ver página 23
dt_final <- read_excel("dt_final.xlsx", sheet="dt_final") # Ver página 59
dtx <- as.data.table(dt_final)
dt_x <- dtx[, paste(collapse=""), by = c("doc_id", "sentence")]
train <- dt_x
stpw <- stopwords(kind = "pt")
docs = tm::Corpus(tm::VectorSource(train$sentence))
docs <- tm_map(docs, removeWords, stpw)
df2 <- ldply(docs, data.frame)
colnames(df2) <- c( "text")
doisgram <- df2 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)
print(tbl_df(doisgram), n=10)
```

```
   bigram               n
   <chr>            <int>
 1 iva touradas       507
 2 redução iva        239
 3 iva é              225
 4 iva 6              210
 5 iva eletricidade   208
 6 baixar iva         192
 7 descida iva        169
 8 fisco é            129
 9 fuga fisco         116
10 6 iva              107
```

```
tresgram <- df2 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 3) %>% dplyr::count(bigram, sort = TRUE)

print(tbl_df(tresgram), n=10)
```

	bigram	n
	<chr>	<int>
1	iva touradas 6	58
2	iva 6 touradas	57
3	descida iva touradas	46
4	imposto sobre valor	46
5	baixar iva touradas	42
6	redução iva touradas	41
7	integridade recepção fisco	37
8	50 mil euros	36
9	fim isenção iva	34
10	iva touradas é	33

```
quatrogram <- df2 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 4) %>% dplyr::count(bigram, sort = TRUE)

print(tbl_df(quatrogram), n=10)
```

	bigram	n
	<chr>	<int>
1	isenção iva artistas tauromáquicos	28
2	imposto sobre valor agregado	27
3	pode ser arquivado apresentação	27
4	fim isenção iva artistas	26
5	vida integridade recepção fisco	24
6	integridade recepção fisco documento	23
7	arco responsabilidade perante fisco	22
8	ser arquivado apresentação fisco	22
9	iva imposto sobre valor	21
10	errei arco responsabilidade perante	20

```
cincogram <- df2 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 5) %>% dplyr::count(bigram, sort = TRUE)

print(tbl_df(cincogram), n=10)
```

	bigram	n
	<chr>	<int>
1	fim isenção iva artistas tauromáquicos	24
2	pode ser arquivado apresentação fisco	21
3	errei arco responsabilidade perante fisco	20
4	ser arquivado apresentação fisco solicitado	17
5	vida integridade recepção fisco documento	17
6	integridade recepção fisco documento eletrônico	15
7	imposto sobre valor agregado iva	13
8	acima leis deve explicações fisco	12
9	iva imposto sobre valor agregado	12

```
10 pan obtém fim isenção iva                          12
```

```
doisgram2 <- doisgram %>% separate(bigram, c("word1", "word2"), sep = " ")
total_words <- doisgram2  %>% group_by(word1) %>% summarize(total = sum(n))
doisgram2 <- doisgram2 %>% bind_tf_idf(word1, word2, n) %>% arrange(desc(tf_idf))
bigramx <- as.data.frame(doisgram2)
head(bigramx, 10)
```

	word1	word2	n	tf	idf	tf_idf
1	integridade	recepção	37	1	9.781998	9.781998
2	setores	produtivos	8	1	9.781998	9.781998
3	coletes	amarelos	7	1	9.781998	9.781998
4	698	259	6	1	9.781998	9.781998
5	927	698	6	1	9.781998	9.781998
6	contacte	927	6	1	9.781998	9.781998
7	lotes	residuais	6	1	9.781998	9.781998
8	brutalidade	confiscatória	5	1	9.781998	9.781998
9	7anos	liberação	4	1	9.781998	9.781998
10	comemoramos	7anos	4	1	9.781998	9.781998

```
trigram <- as.data.frame(tresgram)
tresgram2  <- tresgram %>% separate(bigram, c("word1", "word2", "word3"), sep = " ")
total_words <- tresgram2  %>% group_by(word1) %>% summarize(total = sum(n))
tresgram2 <- tresgram2 %>% bind_tf_idf(word1, word2,  n) %>% arrange(desc(tf_idf))
print(tbl_df(tresgram2), n=10)
```

	word1	word2	word3	n	tf	idf	tf_idf
	<chr>	<chr>	<chr>	<int>	<dbl>	<dbl>	<dbl>
1	integridade	recepção	fisco	37	1	9.78	9.78
2	setores	produtivos	buscando	8	1	9.78	9.78
3	698	259	229	6	1	9.78	9.78
4	927	698	259	6	1	9.78	9.78
5	contacte	927	698	6	1	9.78	9.78
6	brutalidade	confiscatória	fisco	5	1	9.78	9.78
7	comemoramos	7anos	liberação	4	1	9.78	9.78
8	lembraram	cabrõezinhos	grupo	3	1	9.78	9.78
9	1074	retalhistas	regime	2	1	9.78	9.78
10	11a30	outubro18	interessados	2	1	9.78	9.78

 # n-grams – análise evolutiva / temporal

```
nrow(dt_x)
[1] 11464
```

```
twt1$doc_id <- paste("doc", twt1$doc_id, sep = "", collapse = NULL)
```

```
t_datas <- sqldf ("select dt_x.doc_id, dt_x.sentence, twt1.Per_Created from dt_x inner join twt1 where dt_x.doc_id = twt1.doc_id group by dt_x.doc_id, dt_x.sentence, twt1.Per_Created ")

train_201809 <- subset(t_datas, Per_Created == 201809)

train_201812 <- subset(t_datas, Per_Created == 201812)

train_201902 <- subset(t_datas, Per_Created == 201902)


nrow(t_datas)

# [1] 23931

nrow(train_201809)

# [1] 7156

nrow(train_201812)

# [1] 8993

nrow(train_201902)

# [1] 7782


stpw <- stopwords(kind = "pt")

docs09 = tm::Corpus(tm::VectorSource(train_201809$sentence))

docs09 <- tm_map(docs09, removeWords, stpw)

df09 <- ldply(docs09, data.frame)

colnames(df09) <- c( "text")

unigram09 <- df09 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 1) %>% dplyr::count(bigram, sort = TRUE)

unigram09 <- as.data.frame(unigram09)

unigram09$percent <- unigram09$n / sum(unigram09$n)

print(tbl_df(unigram09), n=20)

#  tibble: 13,139 x 3
   bigram           n percent
   <chr>        <int>   <dbl>
 1 iva           4145 0.0572
 2 é             1716 0.0237
 3 fisco         1404 0.0194
 4 irs           1357 0.0187
 5 touradas       824 0.0114
 6 6              552 0.00761
 7 vai            443 0.00611
 8 irc            405 0.00559
 9 sobre          388 0.00535
10 ser            342 0.00472
11 redução        307 0.00423
12 ps             304 0.00419
13 baixar         300 0.00414
14 pagar          282 0.00389
15 eletricidade   273 0.00377
16 governo        265 0.00366
```

```
17 23             249 0.00343
18 ter            246 0.00339
19 taxa           242 0.00334
20 imposto        221 0.00305
```

```
doisgram09 <- df09 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)

doisgram09 <- as.data.frame(doisgram09)

doisgram09$percent <- doisgram09$n / sum(doisgram09$n)

print(tbl_df(doisgram09), n=20)
```

```
# A tibble: 50,424 x 3
    Bigram201809    n   perct
 1 iva touradas    492 6.80e-3
 2 redução iva     221 3.05e-3
 3 iva eletrici~   197 2.72e-3
 4 iva 6           196 2.71e-3
 5 baixar iva      184 2.54e-3
 6 descida iva     156 2.16e-3
 7 iva é           147 2.03e-3
 8 6 iva            96 1.33e-3
 9 isenção iva      70 9.67e-4
10 é iva            69 9.53e-4
11 iva 23           67 9.26e-4
12 iva energia      65 8.98e-4
13 taxa iva         65 8.98e-4
14 touradas 6       65 8.98e-4
15 23 iva           64 8.84e-4
16 6 touradas       64 8.84e-4
17 sobre iva        64 8.84e-4
18 imposto sobre    60 8.29e-4
19 50 irs           59 8.15e-4
20 baixa iva        59 8.15e-4
```

```
docs12 = tm::Corpus(tm::VectorSource(train_201812$sentence))

docs12 <- tm_map(docs12, removeWords, stpw)

df12 <- ldply(docs12, data.frame)

colnames(df12) <- c( "text")

unigram12 <- df12 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 1) %>% dplyr::count(bigram, sort = TRUE)

unigram12 <- as.data.frame(unigram12)

unigram12$percent <- unigram12$n / sum(unigram12$n)

print(tbl_df(unigram12), n=20)
```

```
# A tibble: 15,319 x 3
   bigram           n percent
   <chr>        <int>   <dbl>
```

```
 1 iva           4904 0.0541
 2 é             2293 0.0253
 3 fisco         2085 0.0230
 4 irs           1675 0.0185
 5 touradas       862 0.00951
 6 6              591 0.00652
 7 vai            529 0.00583
 8 irc            523 0.00577
 9 sobre          448 0.00494
10 ser            430 0.00474
11 pagar          358 0.00395
12 redução        335 0.00369
13 ter            323 0.00356
14 baixar         318 0.00351
15 ps             308 0.00340
16 23             297 0.00328
17 governo        296 0.00326
18 eletricidade   281 0.00310
19 taxa           273 0.00301
20 pode           263 0.00290
```

```
doisgram12 <- df12 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)

doisgram12 <- as.data.frame(doisgram12)

doisgram12$percent <- doisgram12$n / sum(doisgram12$n)

print(tbl_df(doisgram12), n=20)
```

```
# A tibble: 62,882 x 3
   bigram               n  percent
   <chr>            <int>    <dbl>
 1 iva touradas       507 0.00559
 2 redução iva        234 0.00258
 3 iva 6              209 0.00230
 4 iva é              200 0.00221
 5 iva eletricidade   198 0.00218
 6 baixar iva         190 0.00210
 7 descida iva        165 0.00182
 8 6 iva              103 0.00114
 9 fisco é             99 0.00109
10 é iva               90 0.000993
11 23 iva              84 0.000926
12 iva 23              80 0.000882
13 fuga fisco          77 0.000849
14 isenção iva         75 0.000827
15 pode ser            71 0.000783
16 sobre iva           71 0.000783
17 imposto sobre       68 0.000750
```

```
18 touradas 6          68 0.000750
19 iva energia         67 0.000739
20 taxa iva            67 0.000739
```

```
docs02 = tm::Corpus(tm::VectorSource(train_201902$sentence))
docs02 <- tm_map(docs02, removeWords, stpw)
df02 <- ldply(docs02, data.frame)
colnames(df02) <- c( "text")
unigram02 <- df02 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 1) %>% dplyr::count(bigram, sort = TRUE)
unigram02 <- as.data.frame(unigram02)
unigram02$percent <- unigram02$n / sum(unigram02$n)
print(tbl_df(unigram02), n=20)
```

```
# A tibble: 13,936 x 3
   bigram           n percent
   <chr>        <int>   <dbl>
 1 iva           4411 0.0560
 2 é             1957 0.0249
 3 fisco         1687 0.0214
 4 irs           1420 0.0180
 5 touradas       839 0.0107
 6 6              561 0.00713
 7 vai            462 0.00587
 8 irc            427 0.00542
 9 sobre          411 0.00522
10 ser            366 0.00465
11 pagar          318 0.00404
12 redução        314 0.00399
13 ps             305 0.00387
14 baixar         303 0.00385
15 ter            279 0.00354
16 governo        278 0.00353
17 eletricidade   276 0.00351
18 23             266 0.00338
19 taxa           246 0.00312
20 imposto        228 0.00290
```

```
doisgram02 <- df02 %>% dplyr::select(text) %>% unnest_tokens(bigram, text, token = "ngrams", n = 2) %>% dplyr::count(bigram, sort = TRUE)
doisgram02 <- as.data.frame(doisgram02)
doisgram02$percent <- doisgram02$n / sum(doisgram02$n)
print(tbl_df(doisgram02), n=20)
```

```
# A tibble: 54,722 x 3
   bigram              n  percent
   <chr>           <int>    <dbl>
```

1	iva touradas	499	0.00634
2	redução iva	223	0.00283
3	iva 6	198	0.00252
4	iva eletricidade	198	0.00252
5	baixar iva	184	0.00234
6	iva é	176	0.00224
7	descida iva	157	0.00199
8	6 iva	102	0.00130
9	é iva	77	0.000978
10	23 iva	76	0.000965
11	fisco é	76	0.000965
12	isenção iva	75	0.000953
13	iva 23	70	0.000889
14	sobre iva	68	0.000864
15	touradas 6	66	0.000838
16	taxa iva	65	0.000826
17	imposto sobre	64	0.000813
18	iva energia	64	0.000813
19	6 touradas	63	0.000800
20	fuga fisco	62	0.00078

ANEXO 3 - CÓDIGO "R" PARA ANÃLISE *POS-TAGGING*/COOCORRÊNCIA

```
# adaptado de https://nballier.github.io/tm4ss.github.io/Tutorial_5_Co-occurrence.html

setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library(udpipe)
library(data.table)
library(tm)
library(ptstem)
library(udpipe)
library(dplyr)
library(purrr)
library("textcat")
library(plyr)
library(lattice)

dt <- read_excel("dt_final.xlsx", sheet="dt_final") # Ver página 59
dtx <- as.data.table(dt)
dff <- dtx[, paste(collapse=""), by = c("doc_id", "sentence")]
dff <- dff[,1:2]
colnames(dff) <- c( "doc_id","text")
udmodel_pt <- udpipe_load_model(file = "portuguese-bosque-ud-2.3-181115.udpipe")
x <- udpipe_annotate(udmodel_pt, dff$text, doc_id = paste("doc", dff$doc_id, sep = ""))
y <- as.data.frame(x)

y$lang <- textcat(y$sentence)
sim_pt <- subset(y, lang == "portuguese")
d_spt <- as.data.table(sim_pt)
d_spt$stempt <- ptstem_words(d_spt$token, algorithm = "porter", complete = T)

d_spt %>% dplyr::count(token, sort=T)
zz2 <- map_df(d_spt, ~.x)
stpw <- stopwords(kind = "pt")
nalfa <- c('.', ',', ';', '?', '!', '…', ':', '-', '%', '(', ')', '/', '...', '“', '”', 'n', 'p', '\"', '+', '&', '€', '–')

zz3 <- zz2[ !(zz2$token %in% stpw), ]
zzz <- zz3[ !(zz3$token %in% nalfa), ]
tt1 <- zzz %>% dplyr::count(token, sort=T)
```

```
print(tt1, n = 10)
```

```
   token        n
   <chr>    <int>
 1 iva       5504
 2 fisco     3158
 3 é         2796
 4 irs       2300
 5 touradas   866
 6 irc        693
 7 vai        637
 8 6          616
 9 ser        560
10 sobre      553
```

```
zz33 <- zz2[ !(zz2$lemma %in% stpw), ]
zzw <- zz33[ !(zz33$lemma %in% nalfa), ]
tt2 <- zzw %>% dplyr::count(lemma, sort=T)
print(tt2, n = 10)
```

```
   lemma       n
   <chr>   <int>
 1 NA      14433
 2 ser      5069
 3 iva      4052
 4 ir       3546
 5 fisco    3157
 6 ter      1779
 7 ivo      1395
 8 pagar    1118
 9 estar    1007
10 tourada   862
```

```
zz333 <- zz2[ !(zz2$stempt %in% stpw), ]
zzy <- zz333[ !(zz333$stempt %in% nalfa), ]
tt3 <- zzy %>% dplyr::count(stempt, sort=T)
print(tt3, n = 10)
```

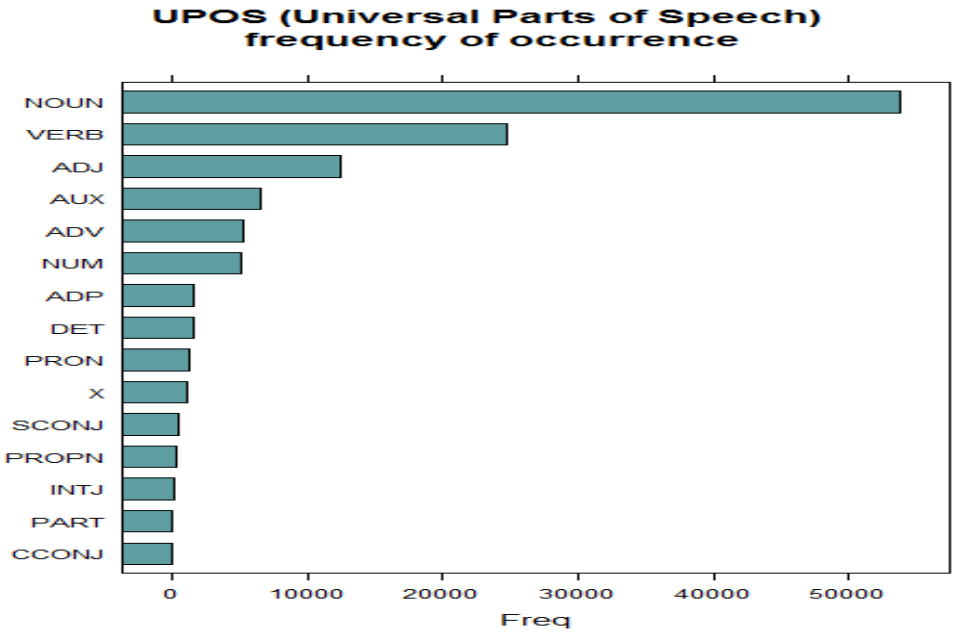
```
   stempt       n
   <chr>    <int>
 1 iva       5508
 2 fisco     3374
 3 é         2796
 4 irs       2300
 5 pagar     1177
 6 touradas  1098
 7 ser        816
```

```
 8 irc        693
 9 baixar     680
10 quer       656
```

```
stats <- txt_freq(zzz$upos)
stats
```

```
     key  freq    freq_pct
1   NOUN 53735 46.91988649
2   VERB 24716 21.58131412
3    ADJ 12357 10.78978389
4    AUX  6502  5.67736302
5    ADV  5225  4.56232264
6    NUM  5106  4.45841519
7    ADP  1654  1.44422615
8    DET  1610  1.40580659
9   PRON  1268  1.10718184
10     X  1055  0.92119625
11 SCONJ   384  0.33529797
12 PROPN   264  0.23051735
13  INTJ   100  0.08731718
14  PART    62  0.05413665
15 CCONJ    13  0.01135123
```

```
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = stats, col = "cadetblue",
        main = "UPOS (Universal Parts of Speech)\n frequency of occurrence",
        xlab = "Freq")
```

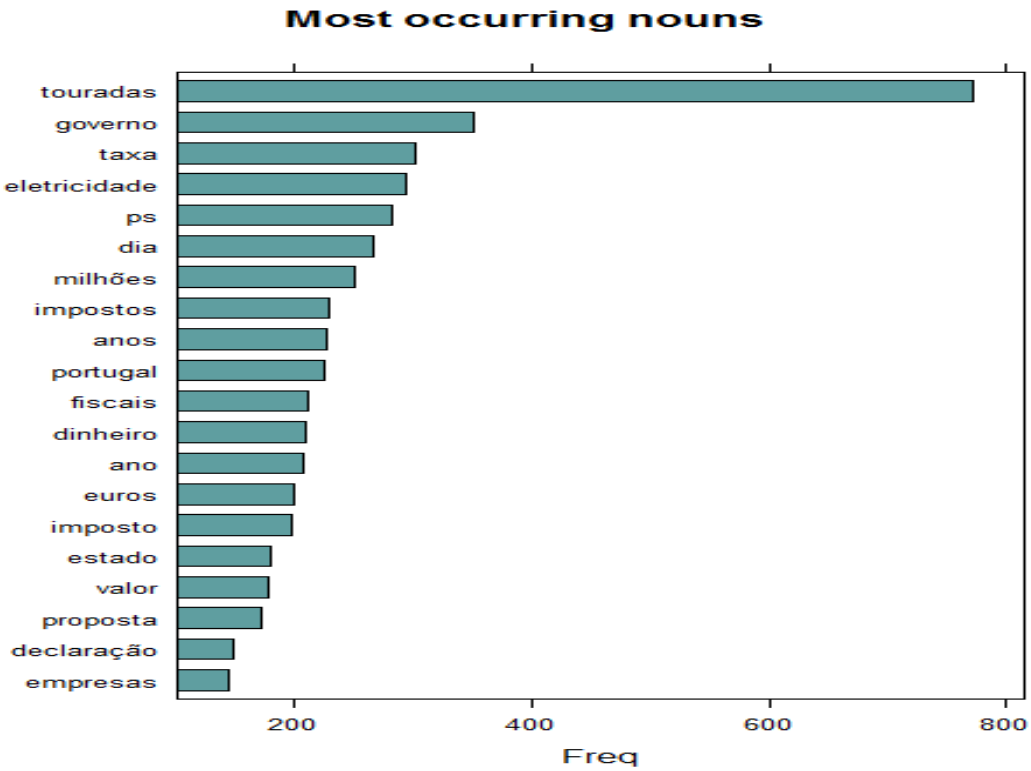


```
zzz$upos <- ifelse(zzz$token == "iva", "NOUN", zzz$upos) # Verificámos que o token “iva” podia assumir NOME, VERBO ou ADJETIVO.
zzz$upos <- ifelse(zzz$token == "redução", "VERB", zzz$upos)
zzz$upos <- ifelse(zzz$token == "descida", "VERB", zzz$upos)
zzz$upos <- ifelse(zzz$token == "fiscais", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "fisco", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "fiscal", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "tributária", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "portugal", "NOUN", zzz$upos)

stats <- txt_freq(zzz$upos)
stats
```

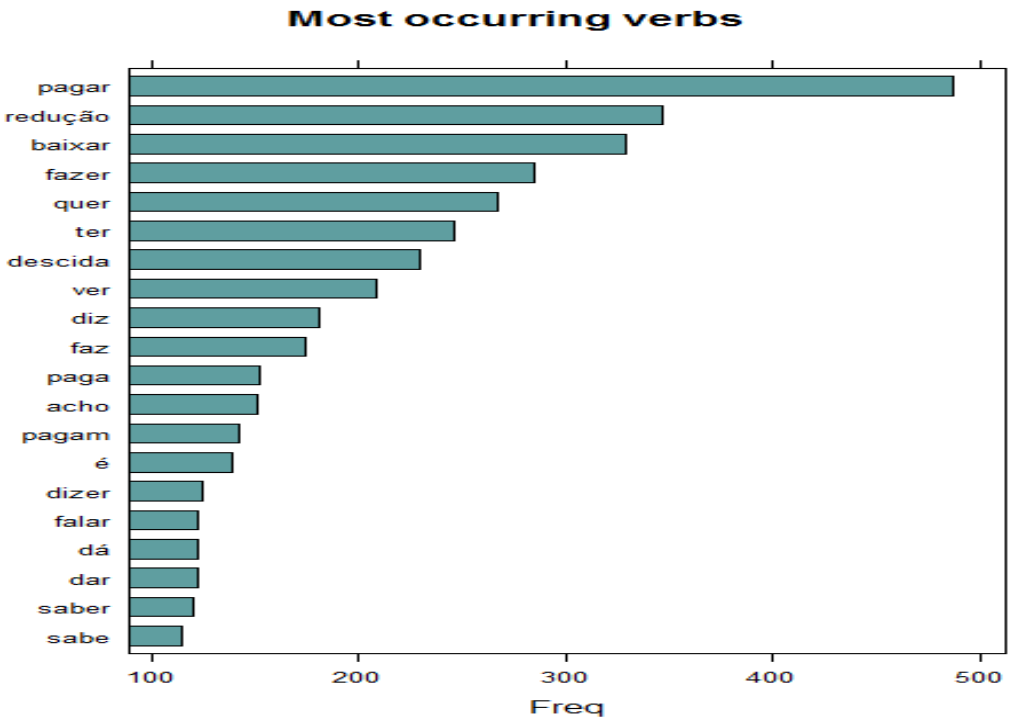
	key	freq	freq_pct
1	NOUN	55458	48.42436149
2	VERB	24978	21.81008513
3	ADJ	10434	9.11067453
4	AUX	6502	5.67736302
5	ADV	5169	4.51342502
6	NUM	5106	4.45841519
7	ADP	1654	1.44422615
8	DET	1610	1.40580659
9	PRON	1268	1.10718184
10	X	1055	0.92119625
11	SCONJ	384	0.33529797
12	PROPN	258	0.22527832
13	INTJ	100	0.08731718
14	PART	62	0.05413665
15	CCONJ	13	0.01135123

```
zz2 <- map_df(d_spt, ~.x)
stpw <- stopwords(kind = "pt")
nalfa <- c('.', ',', ';', '?', '!', '…', ':', '-', '%', '(', ')', '/', '...', '“', '”', 'n', 'p', '\"', '+', '&', '€', '–')
nhash <- c('autoridade tributaria', 'autoridades fiscais ', 'finanças', 'fisco', 'irs', 'iva', 'irc')

zz3 <- zz2[ !(zz2$token %in% stpw), ]
zz4 <- zz3[ !(zz3$token %in% nalfa), ]
zzz <- zz4[ !(zz4$token %in% nhash), ]
zzz$upos <- ifelse(zzz$token == "iva", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "redução", "VERB", zzz$upos)
zzz$upos <- ifelse(zzz$token == "descida", "VERB", zzz$upos)
```

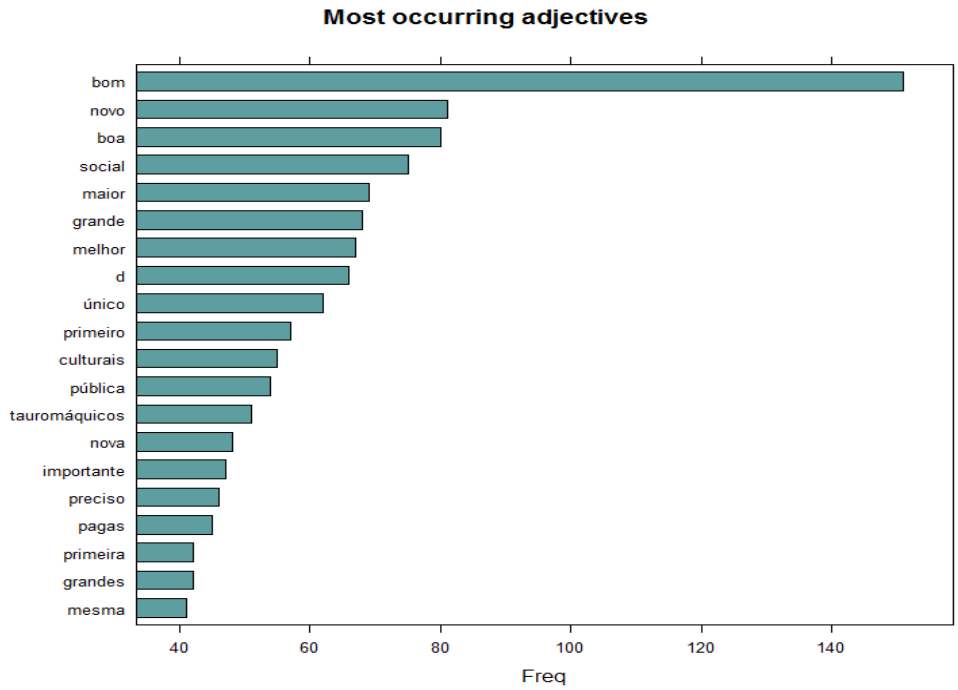
```
zzz$upos <- ifelse(zzz$token == "fiscais", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "fisco", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "fiscal", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "tributária", "NOUN", zzz$upos)
zzz$upos <- ifelse(zzz$token == "portugal", "NOUN", zzz$upos)

stats <- subset(zzz, upos %in% c("NOUN"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
        main = "Most occurring nouns", xlab = "Freq")
```




```
stats <- subset(zzz, upos %in% c("VERB"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
        main = "Most occurring verbs", xlab = "Freq")
```

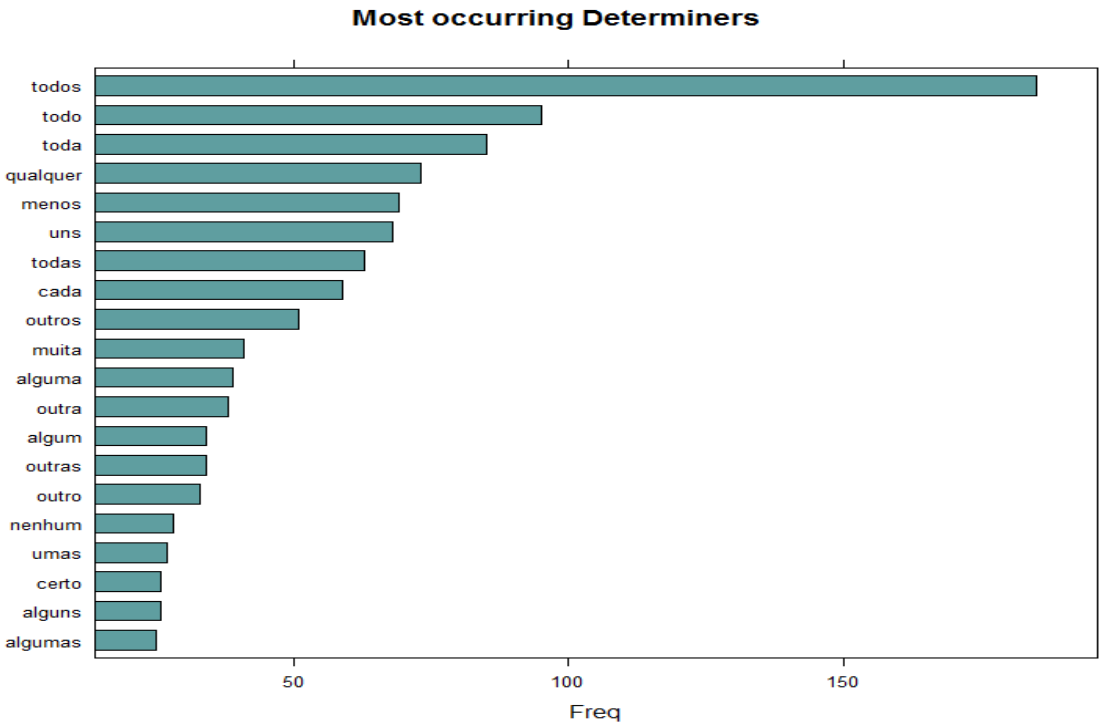



```
stats <- subset(zzz, upos %in% c("ADJ"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
         main = "Most occurring adjectives", xlab = "Freq")
```

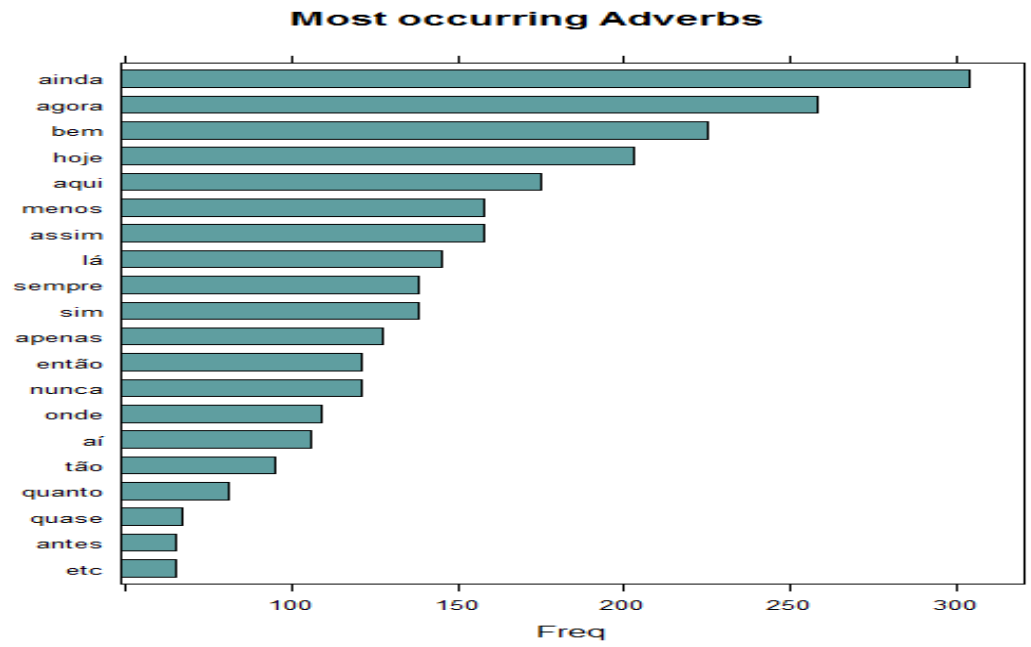


```
stats <- subset(zzz, upos %in% c("AUX"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
         main = "Verbos Auxiliares", xlab = "Freq")
```

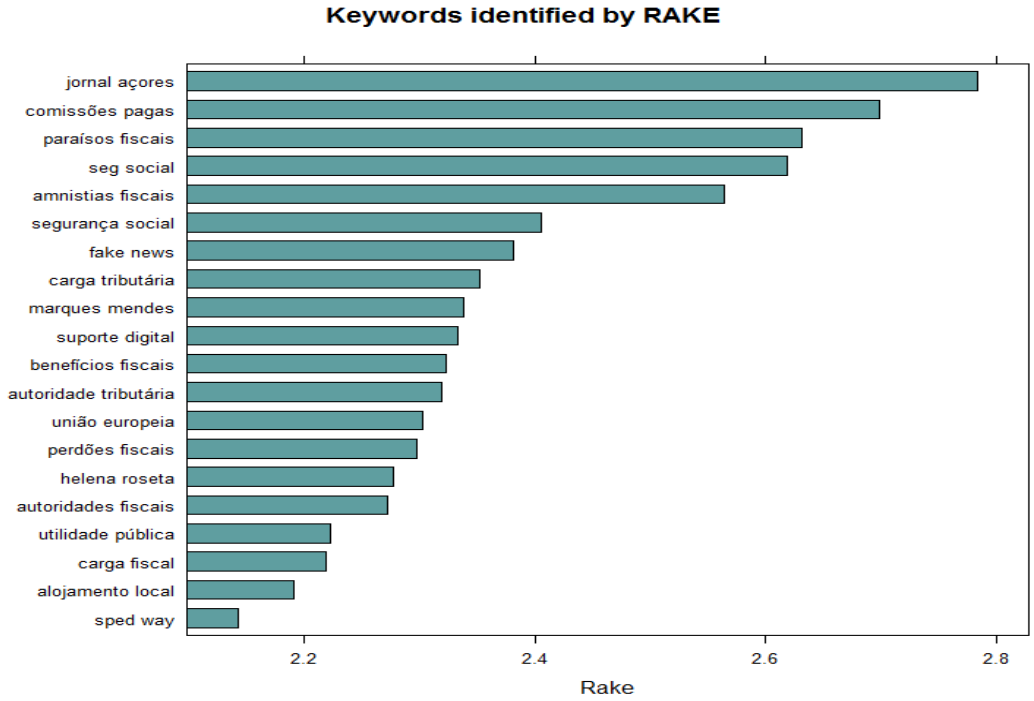
```
stats <- subset(zzz, upos %in% c("DET"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
         main = "Most occurring Determiners", xlab = "Freq")
```




```
stats <- subset(zzz, upos %in% c("ADV"))
stats <- txt_freq(stats$token)
stats$key <- factor(stats$key, levels = rev(stats$key))
barchart(key ~ freq, data = head(stats, 20), col = "cadetblue",
         main = "Most occurring Adverbs", xlab = "Freq")
```




```
stats <- keywords_rake(x = y, term = "token", group = "doc_id",
                       relevant = y$upos %in% c("NOUN", "ADJ"))
stats$key <- factor(stats$keyword, levels = rev(stats$keyword))
barchart(key ~ rake, data = head(subset(stats, freq > 3), 20), col = "cadetblue",
       main = "Keywords identified by RAKE",
       xlab = "Rake")
```



```
cooc <- cooccurrence(x = subset(zzz, upos %in% c("NOUN", "ADJ")),
                     term = "token",
                     group = c("sentence"))
head(cooc, 10)
```

```
          term1    term2 cooc
1           iva touradas  829
2  eletricidade      iva  331
3           iva       ps  271
4       governo      iva  195
5           iva     taxa  191
6         fisco  milhões  165
7           dia      iva  153
8       imposto      iva  153
9            ps touradas  153
10          iva proposta  146
```

```
cooc <- cooccurrence(x = subset(zzz, upos %in% c("NOUN", "VERB")),
                     term = "stempt",
                     group = c("sentence"))
head(cooc,10)
```

```
          term1    term2 cooc
1           iva touradas 1164
2        baixar      iva  546
3           iva    pagar  537
4       descida      iva  361
5       imposto      iva  352
6           iva     quer  342
7  eletricidade      iva  331
8           iva  redução  298
9         fisco    pagar  280
10          irs    pagar  273
```

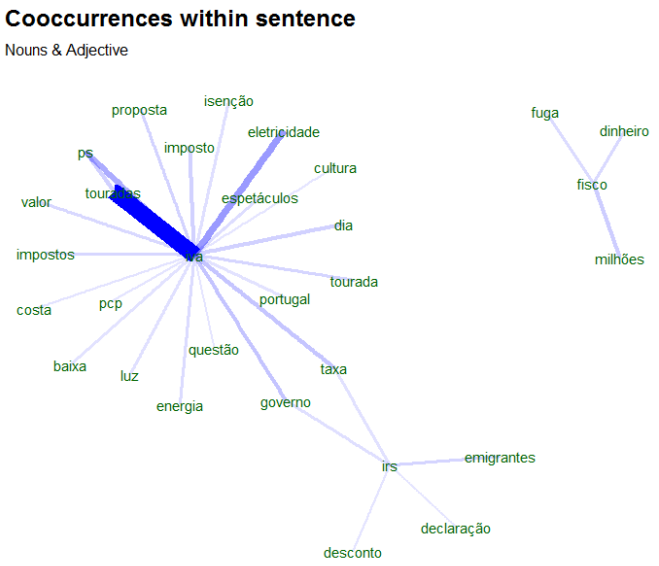
```
cooc <- cooccurrence(x = subset(zzz, upos %in% c("VERB", "ADJ")),
                     term = "stempt",
                     group = c("sentence"))
head(cooc,10)
```

```
      term1 term2 cooc
1     pagar  quer   75
2    baixar  quer   56
3       diz  quer   47
4     pagar   ver   47
5      quer saber   44
```

```
6      fazer pagar   41
7      pagar saber   41
8       deve pagar   38
9      fazer  quer   34
10 aprender fazer   31
```

 # Co-occurrence analysis / GRAFO

```
library(igraph)
library(ggraph)
library(ggplot2)

cooc <- cooccurrence(x = subset(zzz, upos %in% c("NOUN", "ADJ")),
                     term = "token",
                     group = c("sentence"))

wordnetwork <- head(cooc, 30)
wordnetwork <- graph_from_data_frame(wordnetwork)
ggraph(wordnetwork, layout = "fr") +
  geom_edge_link(aes(width = cooc, edge_alpha = cooc), edge_colour = "blue") +
  geom_node_text(aes(label = name), col = "darkgreen", size = 4) +
  theme_graph(base_family = "Arial Narrow") +
  theme(legend.position = "none") +
  labs(title = "Cooccurrences within sentence", subtitle = "Nouns & Adjective")
```



```
cooc <- cooccurrence(zzz$token, relevant = zzz$upos %in% c("NOUN", "ADJ"), skipgram = 1)
head(cooc,10)
```

```
      term1        term2 cooc
1       iva     touradas  574
2       iva eletricidade  237
3       iva          iva  124
4      taxa          iva  113
5      fuga        fisco  109
6       iva      energia   93
7   isenção          iva   86
8  desconto          irs   85
9       irs   emigrantes   75
10  milhões        fisco   71
```

```
dtx <- as.data.table(zzz)
dt_x <- dtx[, paste(collapse=""), by = c("doc_id", "sentence")]
dt_x$sentence <- gsub("[[:punct:]]", " ", dt_x$sentence)
dt_x$sentence <- gsub("(?<=[\\s])\\s*|^\\s+|\\s+$", "", dt_x$sentence, perl=TRUE) # remover
                                                                    # espaços em branco extra
train <- dt_x[,1:2]
nrow(train)
[1] 11464

stpw <- stopwords(kind = "pt")
docs = tm::Corpus(tm::VectorSource(train$sentence))
docs <- tm_map(docs, removeWords, stpw)

minimumFrequency <- 10
binDTM <- DocumentTermMatrix(docs, control=list(bounds = list(global=c(minimumFrequency,
Inf)), weighting = weightBin))
require(Matrix)
binDTM <- sparseMatrix(i = binDTM$i, j = binDTM$j, x = binDTM$v, dims = c(binDTM$nrow,
binDTM$ncol), dimnames = dimnames(binDTM))
coocCounts <- t(binDTM) %*% binDTM
as.matrix(coocCounts[1:350, 1:350])

coocTerm <- "iva"
k <- nrow(binDTM)
ki <- sum(binDTM[, coocTerm])
kj <- colSums(binDTM)
```

```
names(kj) <- colnames(binDTM)
kij <- coocCounts[coocTerm, ]
mutualInformationSig <- log(k * kij / (ki * kj))
mutualInformationSig <- mutualInformationSig[order(mutualInformationSig, decreasing = TRUE)]
dicesig <- 2 * kij / (ki + kj)
dicesig <- dicesig[order(dicesig, decreasing=TRUE)]
logsig <- 2 * ((k * log(k)) - (ki * log(ki)) - (kj * log(kj)) + (kij * log(kij))
        + (k - ki - kj + kij) * log(k - ki - kj + kij)
        + (ki - kij) * log(ki - kij) + (kj - kij) * log(kj - kij)
        - (k - ki) * log(k - ki) - (k - kj) * log(k - kj))
logsig <- logsig[order(logsig, decreasing=T)]
# Put all significance statistics in one Data-Frame
resultOverView <- data.frame(
  names(sort(kij, decreasing=T)[1:10]), sort(kij, decreasing=T)[1:10],
  names(mutualInformationSig[1:10]), mutualInformationSig[1:10],
  names(dicesig[1:10]), dicesig[1:10],
  names(logsig[1:10]), logsig[1:10],
  row.names = NULL)
colnames(resultOverView) <- c("Freq-terms", "Freq", "MI-terms", "MI", "Dice-Terms", "Dice", "LL-Terms",
"LL")
print(resultOverView)
```

	Freq-terms	Freq	MI-terms	MI	Dice-Terms	Dice	LL-Terms	LL
1	iva	5269	iva	0.7773711	iva	1.00000000	fisco	4750.3910
2	touradas	814	restaurantes	0.7773711	touradas	0.26727959	irs	2341.8025
3	vai	313	eletricidade	0.7773711	vai	0.10662579	touradas	1257.7596
4	sobre	293	periódica	0.7773711	eletricidade	0.10501708	irc	518.4211
5	eletricidade	292	restauração	0.7773711	sobre	0.10115657	baixar	217.1407
6	redução	277	electricidade	0.7773711	redução	0.09873463	fiscais	198.6713
7	baixar	270	toureiros	0.7773711	baixar	0.09667025	redução	183.2432
8	ser	242	hospital	0.7773711	ser	0.08340514	emigrantes	160.9676
9	imposto	195	madre	0.7773711	imposto	0.07032095	espetáculos	151.5520
10	pagar	194	implementação	0.7773711	pagar	0.06758404	descida	140.4419

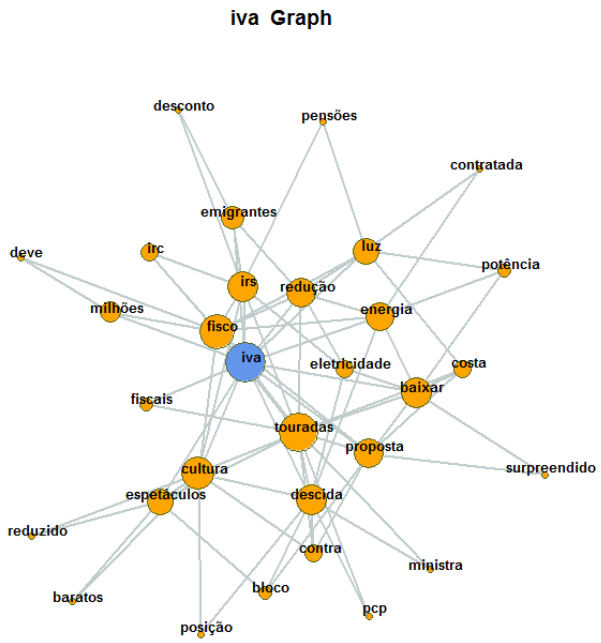
```
# Ficheiro “calculateCoocStatistics.R" tem que ser colocado na pasta de trabalho

# Read in the source code for the co-occurrence calculation
source("calculateCoocStatistics.R")
# Definition of a parameter for the representation of the co-occurrences of a concept
numberOfCoocs <- 15
# Determination of the term of which co-competitors are to be measured.
coocTerm <- "iva"
coocs <- calculateCoocStatistics(coocTerm, binDTM, measure="LOGLIK")
print(coocs[1:numberOfCoocs])
```

```
      fisco         irs    touradas         irc      baixar     fiscais     redução  emigrantes espetáculos
descida
 4750.39096  2341.80251  1257.75963   518.42114   217.14066   198.67132   183.24322   160.96765   151.55205
140.44191
    energia     milhões         luz     cultura    proposta
  135.71379   121.44557   117.94912   114.89182    91.66245
```

```
resultGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
# The structure of the temporary graph object is equal to that of the resultGraph
tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
# Fill the data.frame to produce the correct number of lines
tmpGraph[1:numberOfCoocs, 3] <- coocs[1:numberOfCoocs]
# Entry of the search word into the first column in all lines
tmpGraph[, 1] <- coocTerm
# Entry of the co-occurrences into the second column of the respective line
tmpGraph[, 2] <- names(coocs)[1:numberOfCoocs]
# Set the significances
tmpGraph[, 3] <- coocs[1:numberOfCoocs]
# Attach the triples to resultGraph
resultGraph <- rbind(resultGraph, tmpGraph)
# Iteration over the most significant numberOfCoocs co-occurrences of the search term
for (i in 1:numberOfCoocs){newCoocTerm <- names(coocs)[i]
coocs2 <- calculateCoocStatistics(newCoocTerm, binDTM, measure="LOGLIK")
coocs2[1:10]
  tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
  tmpGraph[1:numberOfCoocs, 3] <- coocs2[1:numberOfCoocs]
  tmpGraph[, 1] <- newCoocTerm
  tmpGraph[, 2] <- names(coocs2)[1:numberOfCoocs]
  tmpGraph[, 3] <- coocs2[1:numberOfCoocs]
  resultGraph <- rbind(resultGraph, tmpGraph[2:length(tmpGraph[, 1]), ])
}
resultGraph[sample(nrow(resultGraph), 10), ]
           from           to        sig
212     milhões        fisco  149.56792
136     fiscais       estado   26.52999
1212    milhões        dever   26.60910
21        fisco          irs 1403.66852
1410    descida       função   31.00270
45       baixar eletricidade   75.10487
48   emigrantes       metade  103.94741
1315   proposta        bloco   23.92120
914     cultura     reduzido   16.06114
1115   proposta        costa   36.42006
```

```
require(igraph)
```

```
# Set the graph and type. In this case, "F" means "Force Directed"
graphNetwork <- graph.data.frame(resultGraph, directed = F)
# Identification of all nodes with less than 2 edges
graphVs <- V(graphNetwork)[degree(graphNetwork) < 2]
# These edges are removed from the graph
graphNetwork <- delete.vertices(graphNetwork, graphVs)
# Assign colors to edges and nodes (searchterm blue, rest orange)
V(graphNetwork)$color <- ifelse(V(graphNetwork)$name == coocTerm, 'cornflowerblue', 'orange')
# Edges with a significance of at least 50% of the maximum sig- nificance in the graph are drawn in orange
halfMaxSig <- max(E(graphNetwork)$sig) * 0.5
E(graphNetwork)$color <- ifelse(E(graphNetwork)$sig > halfMaxSig, "coral", "azure3")
# Disable edges with radius
E(graphNetwork)$curved <- 0
# Size the nodes by their degree of networking
V(graphNetwork)$size <- log(degree(graphNetwork)) * 5
# All nodes must be assigned a standard minimum-size
V(graphNetwork)$size[V(graphNetwork)$size < 5] <- 3
# edge thickness
E(graphNetwork)$width <- 2
# Define the frame and spacing for the plot
par(mai=c(0,0,1,0))
# Finaler Plot
plot(graphNetwork,
    layout = layout.fruchterman.reingold,  # Force Directed Layout
    main = paste(coocTerm, ' Graph'),
    vertex.label.family = "sans",
    vertex.label.cex = 0.8,
    vertex.shape = "circle",
    vertex.label.dist = 0.5,           # Labels of the nodes moved slightly
    vertex.frame.color = 'darkolivegreen',
    vertex.label.color = 'black',      # Color of node names
    vertex.label.font = 2,         # Font of node names
    vertex.label = V(graphNetwork)$name,       # node names
    vertex.label.cex = 1 # font size of node names
)
```

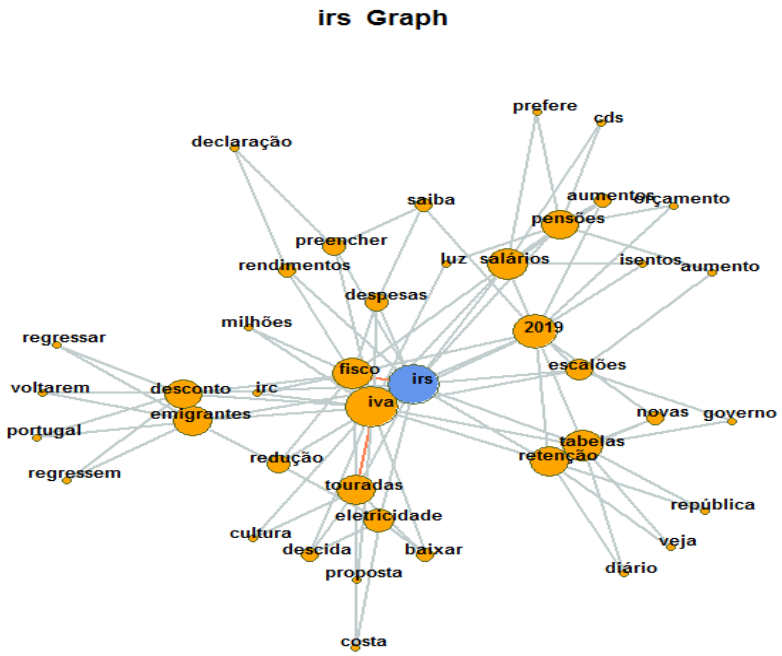


coocTerm <- "irs"

```
k <- nrow(binDTM)
ki <- sum(binDTM[, coocTerm])
kj <- colSums(binDTM)
names(kj) <- colnames(binDTM)
kij <- coocCounts[coocTerm, ]
mutualInformationSig <- log(k * kij / (ki * kj))
mutualInformationSig <- mutualInformationSig[order(mutualInformationSig, decreasing = TRUE)]
dicesig <- 2 * kij / (ki + kj)
dicesig <- dicesig[order(dicesig, decreasing=TRUE)]
logsig <- 2 * ((k * log(k)) - (ki * log(ki)) - (kj * log(kj)) + (kij * log(kij))
        + (k - ki - kj + kij) * log(k - ki - kj + kij)
        + (ki - kij) * log(ki - kij) + (kj - kij) * log(kj - kij)
        - (k - ki) * log(k - ki) - (k - kj) * log(k - kj))
logsig <- logsig[order(logsig, decreasing=T)]
resultOverView <- data.frame(
  names(sort(kij, decreasing=T)[1:10]), sort(kij, decreasing=T)[1:10],
  names(mutualInformationSig[1:10]), mutualInformationSig[1:10],
  names(dicesig[1:10]), dicesig[1:10],
  names(logsig[1:10]), logsig[1:10],
  row.names = NULL)
colnames(resultOverView) <- c("Freq-terms", "Freq", "MI-terms", "MI", "Dice-Terms", "Dice", "LL-Terms",
"LL")
```

```
source("calculateCoocStatistics.R")
numberOfCoocs <- 15
coocTerm <- "irs"
coocs <- calculateCoocStatistics(coocTerm, binDTM, measure="LOGLIK")
resultGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
tmpGraph[1:numberOfCoocs, 3] <- coocs[1:numberOfCoocs]
tmpGraph[, 1] <- coocTerm
tmpGraph[, 2] <- names(coocs)[1:numberOfCoocs]
tmpGraph[, 3] <- coocs[1:numberOfCoocs]
resultGraph <- rbind(resultGraph, tmpGraph)
for (i in 1:numberOfCoocs){newCoocTerm <- names(coocs)[i]
coocs2 <- calculateCoocStatistics(newCoocTerm, binDTM, measure="LOGLIK")
coocs2[1:10]
  tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
  tmpGraph[1:numberOfCoocs, 3] <- coocs2[1:numberOfCoocs]
  tmpGraph[, 1] <- newCoocTerm
  tmpGraph[, 2] <- names(coocs2)[1:numberOfCoocs]
  tmpGraph[, 3] <- coocs2[1:numberOfCoocs]
  resultGraph <- rbind(resultGraph, tmpGraph[2:length(tmpGraph[, 1]), ])
}

require(igraph)
graphNetwork <- graph.data.frame(resultGraph, directed = F)
graphVs <- V(graphNetwork)[degree(graphNetwork) < 2]
graphNetwork <- delete.vertices(graphNetwork, graphVs)
V(graphNetwork)$color <- ifelse(V(graphNetwork)$name == coocTerm, 'cornflowerblue', 'orange')
halfMaxSig <- max(E(graphNetwork)$sig) * 0.5
E(graphNetwork)$color <- ifelse(E(graphNetwork)$sig > halfMaxSig, "coral", "azure3")
E(graphNetwork)$curved <- 0
V(graphNetwork)$size <- log(degree(graphNetwork)) * 5
V(graphNetwork)$size[V(graphNetwork)$size < 5] <- 3
E(graphNetwork)$width <- 2
par(mai=c(0,0,1,0))
plot(graphNetwork,
    layout = layout.fruchterman.reingold,  # Force Directed Layout
    main = paste(coocTerm, ' Graph'),
    vertex.label.family = "Courier",
    vertex.label.cex = 0.8,
    vertex.shape = "circle",
    vertex.label.dist = 0.5,          # Labels of the nodes moved slightly
    vertex.frame.color = 'darkolivegreen',
```

```
    vertex.label.color = 'black',      # Color of node names
    vertex.label.font = 2,         # Font of node names
    vertex.label = V(graphNetwork)$name,       # node names
    vertex.label.cex = 1 # font size of node names
)
```

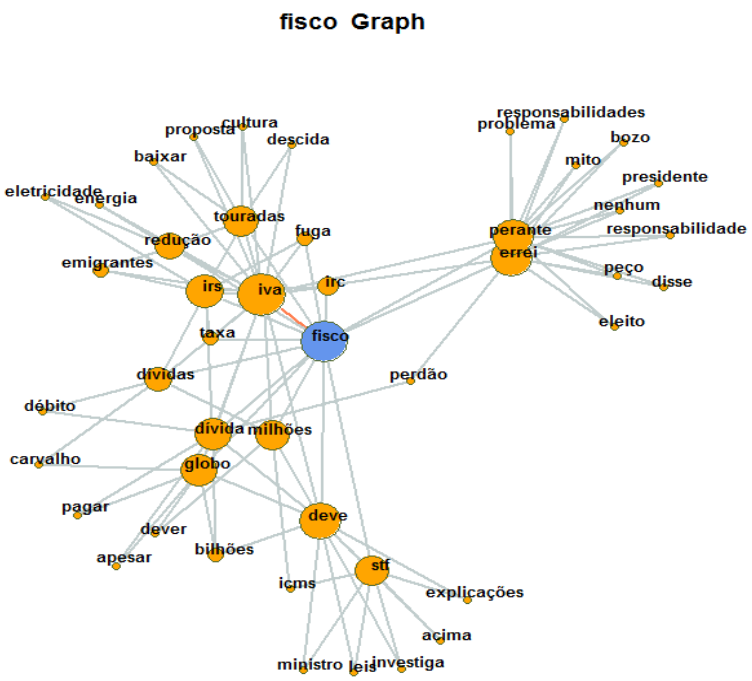


coocTerm <- "fisco"

```
k <- nrow(binDTM)
ki <- sum(binDTM[, coocTerm])
kj <- colSums(binDTM)
names(kj) <- colnames(binDTM)
kij <- coocCounts[coocTerm, ]
mutualInformationSig <- log(k * kij / (ki * kj))
mutualInformationSig <- mutualInformationSig[order(mutualInformationSig, decreasing = TRUE)]
dicesig <- 2 * kij / (ki + kj)
dicesig <- dicesig[order(dicesig, decreasing=TRUE)]
logsig <- 2 * ((k * log(k)) - (ki * log(ki)) - (kj * log(kj)) + (kij * log(kij))
        + (k - ki - kj + kij) * log(k - ki - kj + kij)
        + (ki - kij) * log(ki - kij) + (kj - kij) * log(kj - kij)
        - (k - ki) * log(k - ki) - (k - kj) * log(k - kj))
logsig <- logsig[order(logsig, decreasing=T)]
```

```
resultOverView <- data.frame(
  names(sort(kij, decreasing=T)[1:10]), sort(kij, decreasing=T)[1:10],
  names(mutualInformationSig[1:10]), mutualInformationSig[1:10],
  names(dicesig[1:10]), dicesig[1:10],
  names(logsig[1:10]), logsig[1:10],
  row.names = NULL)
colnames(resultOverView) <- c("Freq-terms", "Freq", "MI-terms", "MI", "Dice-Terms", "Dice", "LL-Terms", "LL")

source("calculateCoocStatistics.R")
numberOfCoocs <- 15
coocTerm <- "fisco"
coocs <- calculateCoocStatistics(coocTerm, binDTM, measure="LOGLIK")
resultGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
tmpGraph[1:numberOfCoocs, 3] <- coocs[1:numberOfCoocs]
tmpGraph[, 1] <- coocTerm
tmpGraph[, 2] <- names(coocs)[1:numberOfCoocs]
tmpGraph[, 3] <- coocs[1:numberOfCoocs]
resultGraph <- rbind(resultGraph, tmpGraph)
for (i in 1:numberOfCoocs){newCoocTerm <- names(coocs)[i]
coocs2 <- calculateCoocStatistics(newCoocTerm, binDTM, measure="LOGLIK")
coocs2[1:10]
  tmpGraph <- data.frame(from = character(), to = character(), sig = numeric(0))
  tmpGraph[1:numberOfCoocs, 3] <- coocs2[1:numberOfCoocs]
  tmpGraph[, 1] <- newCoocTerm
  tmpGraph[, 2] <- names(coocs2)[1:numberOfCoocs]
  tmpGraph[, 3] <- coocs2[1:numberOfCoocs]
  resultGraph <- rbind(resultGraph, tmpGraph[2:length(tmpGraph[, 1]), ])
}

require(igraph)

graphNetwork <- graph.data.frame(resultGraph, directed = F)
graphVs <- V(graphNetwork)[degree(graphNetwork) < 2]
graphNetwork <- delete.vertices(graphNetwork, graphVs)
V(graphNetwork)$color <- ifelse(V(graphNetwork)$name == coocTerm, 'cornflowerblue', 'orange')
halfMaxSig <- max(E(graphNetwork)$sig) * 0.5
E(graphNetwork)$color <- ifelse(E(graphNetwork)$sig > halfMaxSig, "coral", "azure3")
E(graphNetwork)$curved <- 0
V(graphNetwork)$size <- log(degree(graphNetwork)) * 5
V(graphNetwork)$size[V(graphNetwork)$size < 5] <- 3
```

```
E(graphNetwork)$width <- 2
par(mai=c(0,0,1,0))
plot(graphNetwork,
    layout = layout.fruchterman.reingold,  # Force Directed Layout
    main = paste(coocTerm, ' Graph'),
    vertex.label.family = "Courier",
    vertex.label.cex = 0.8,
    vertex.shape = "circle",
    vertex.label.dist = 0.5,          # Labels of the nodes moved slightly
    vertex.frame.color = 'darkolivegreen',
    vertex.label.color = 'black',      # Color of node names
    vertex.label.font = 2,         # Font of node names
    vertex.label = V(graphNetwork)$name,       # node names
    vertex.label.cex = 1 # font size of node names
)
```



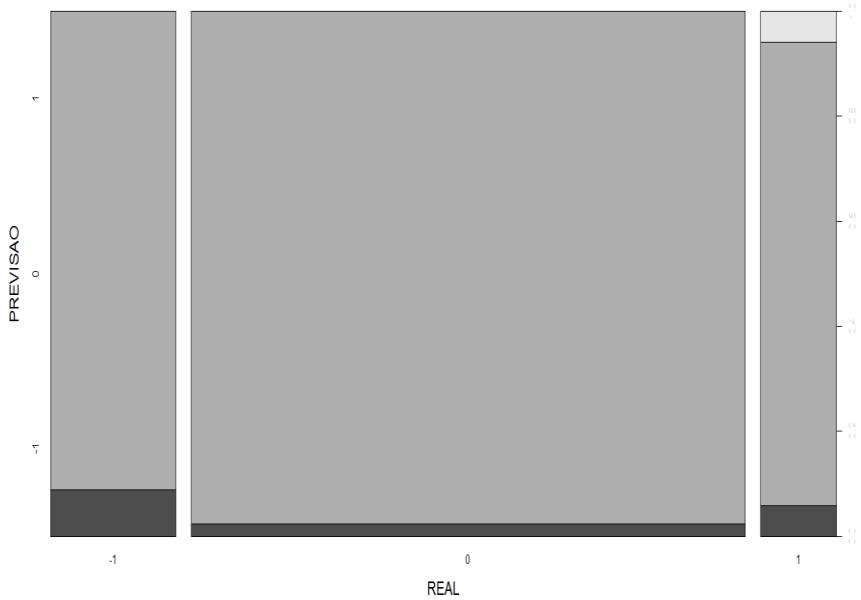
ANEXO 4 - CÓDIGO "R" PARA GERAÇÃO DE MODELOS "*NAIVE BAYES*"

```
setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library("data.table")
library(dplyr)
library(tm)
library(RTextTools)
library(e1071)
library(stats)
library(caret)
library(Rfacebook)
library(tidyverse)
library(ggExtra)
library(magrittr)
library(lubridate)
library(stringr)
library(tidytext)
library(lexiconPT)



am_class <- read_excel("amostra_class1.xlsx", sheet="amostra_class") # Ver página 32
am_tema <- am_class %>%  select(sentence, Opiniao)

set.seed(1)
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
df$class <- as.factor(df$Opiniao)
df$id <- seq.int(nrow(df))
df %>% mutate(id = row_number())


stp <- c("o", "de", "a", "que", "e", "em", "do", "para", "os", "ao", "da", "com", "uma", "por", "na", "as", "um",
"se", "como", "dos", "das", "no", "mas", "me")


corpus <- Corpus(VectorSource(df$sentence))
corpus.clean <- corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
```

```
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords(kind="pt")) %>%
  tm_map(removeWords, stp) %>%
  tm_map(stripWhitespace)
dtm <- DocumentTermMatrix(corpus.clean)

df.train <- df[1:677,]
df.test <- df[678:1015,]
dtm.train <- dtm[1:677,]
dtm.test <- dtm[678:1015,]
corpus.clean.train <- corpus.clean[1:677]
corpus.clean.test <- corpus.clean[678:1015]
f_freq <- findFreqTerms(dtm.train, 3)
dtm.train.nb <- DocumentTermMatrix(corpus.clean.train, control=list(dictionary = f_freq))
dtm.test.nb <- DocumentTermMatrix(corpus.clean.test, control=list(dictionary = f_freq))

convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

trainNB <- apply(dtm.train.nb, 2, convert_count)
testNB <- apply(dtm.test.nb, 2, convert_count)

classifier <- naive*Bayes*(trainNB, df.train$class, laplace = 1)
pred <- predict(classifier, newdata=testNB)
conf.mat <- confusionMatrix(pred, df.test$class)
conf.mat
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1   5   6   2
        0   51 242  30
        1    0   0   2

Overall Statistics

               Accuracy : 0.7367
                 95% CI : (0.6863, 0.7829)
```

```
    No Information Rate : 0.7337
    P-Value [Acc > NIR] : 0.4793

                  Kappa : 0.0978
 Mcnemar's Test P-Value : 1.445e-14

Statistics by Class:

                     Class: -1 Class: 0 Class: 1
Sensitivity            0.08929   0.9758 0.058824
Specificity            0.97163   0.1000 1.000000
Pos Pred Value         0.38462   0.7492 1.000000
Neg Pred Value         0.84308   0.6000 0.904762
Prevalence             0.16568   0.7337 0.100592
Detection Rate         0.01479   0.7160 0.005917
Detection Prevalence   0.03846   0.9556 0.005917
Balanced Accuracy      0.53046   0.5379 0.529412
```

```
tt1 <- data.frame(df.test$class, pred)
par(cex.lab=1.5)
plot(tt1, xlab= "REAL", ylab="PREVISÃO", cex.lab=2)
axis(4, col = "white", col.axis = "white", lwd = 0)
```



ANEXO 5 - CÓDIGO "R" PARA GERAÇÃO DE MODELOS "LEXICON-PT"

```
setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library("data.table")
library(Rfacebook)
library(tidyverse)
library(ggExtra)
library(magrittr)
library(lubridate)
library(stringr)
library(tidytext)
library(lexiconPT)
library(caret)

am_class <- read_excel("amostra_class1.xlsx", sheet="amostra_class") # Ver página 32
am_tema <- am_class %>%  select(sentence, Opiniao)
nrow(am_tema)
# [1] 1015

data("oplexicon_v3.0") # dataset de léxicos
data("sentiLex_lem_PT02") # dataset de léxicos
op30 <- oplexicon_v3.0
sent <- sentiLex_lem_PT02
glimpse(op30)

set.seed(1)
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
glimpse(df)
df$class <- as.factor(df$Opiniao)
df$id <- seq.int(nrow(df))
str(df)
df %>% mutate(id = row_number())
df11 <- df %>% mutate(id = row_number())
df111 <- df11 %>% dplyr::select(id)
nrow(df111)
# [1] 1015
```

```
df_unnested <- df %>% unnest_tokens(term, sentence)
```

```
df_unnested %>% dplyr::select(id, term) %>% head(20)
```

```
# A tibble: 20 x 2
      id term
   <int> <chr>
 1     1 sobre
 2     1 este
 3     1 pros
 4     1 e
 5     1 contras
 6     1 vou
 7     1 apenas
 8     1 dizer
 9     1 isto
10     1 os
11     1 bombeiros
12     1 pagam
13     1 o
14     1 equipamento
15     1 com
16     1 23
17     1 de
18     1 iva
19     1 para
20     1 andarem
```

```
df_unnested %>% left_join(op30, by = "term") %>% left_join(sent %>% dplyr::select(term, lex_polarity = polarity), by = "term") %>% dplyr::select(id, term, polarity, lex_polarity, class) %>% head(10)
```

```
# A tibble: 10 x 5
      id term    polarity lex_polarity class
   <int> <chr>      <int>        <dbl> <fct>
 1     1 sobre         NA           NA 0
 2     1 este          NA           NA 0
 3     1 pros          NA           NA 0
 4     1 e             NA           NA 0
 5     1 contras       NA           NA 0
 6     1 vou           NA           NA 0
 7     1 apenas        NA           NA 0
 8     1 dizer          0           NA 0
 9     1 isto          NA           NA 0
10     1 os            NA           NA 0
```

```
# Vamos então manter no dataframe apenas as palavras que possuem polaridade tanto no ---
```

```
# … OpLexicon como no SentiLex:


df_unnested_2 <- df_unnested %>% inner_join(op30, by = "term") %>% inner_join(sent %>%
dplyr::select(term, lex_polarity = polarity), by = "term") %>% group_by(id, class) %>%
summarise(comment_sentiment_op = sum(polarity), comment_sentiment_lex =
sum(lex_polarity), n_words = n()) %>% ungroup() %>% rowwise() %>% mutate(most_neg =
min(comment_sentiment_lex, comment_sentiment_op),most_pos =
max(comment_sentiment_lex, comment_sentiment_op))


head(df_unnested_2)

nrow(df_unnested_2)

# [1] 267  # Existem 748 tweets que não foram classificados no modelo


table("opLex_Sent" = df_unnested_2$comment_sentiment_op, "class_manual"=
df_unnested_2$class)

          class_manual
opLex_Sent -1  0  1
        -3  1  1  0
        -2  5  6  2
        -1 24 67  7
        0  14 43  8
        1  13 57  7
        2   0 12  0


df_unnested_2$comment_sentiment_op1 <-
as.factor(ifelse(df_unnested_2$comment_sentiment_op > 1, 1,
ifelse(df_unnested_2$comment_sentiment_op < -1, -1, 0)))

df_unnested_2$comment_sentiment_lex1 <-
as.factor(ifelse(df_unnested_2$comment_sentiment_lex > 1, 1,
ifelse(df_unnested_2$comment_sentiment_lex < -1, -1, 0)))

table("opLex_Sent" = df_unnested_2$comment_sentiment_op1, "class_manual"=
df_unnested_2$class)

          class_manual
opLex_Sent  -1   0   1
        -1   6   7   2
        0   51 167  22
        1    0  12   0


conf.mat <- confusionMatrix(df_unnested_2$comment_sentiment_op1, df_unnested_2$class)

conf.mat

Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1   6   7   2
        0   51 167  22
```

```
        1    0  12   0
Overall Statistics
               Accuracy : 0.6479
                 95% CI : (0.5874, 0.7052)
    No Information Rate : 0.6966
    P-Value [Acc > NIR] : 0.9624
                  Kappa : 0.016
 Mcnemar's Test P-Value : 2.418e-08
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity            0.10526  0.89785  0.00000
Specificity            0.95714  0.09877  0.95062
Pos Pred Value         0.40000  0.69583  0.00000
Neg Pred Value         0.79762  0.29630  0.90588
Prevalence             0.21348  0.69663  0.08989
Detection Rate         0.02247  0.62547  0.00000
Detection Prevalence   0.05618  0.89888  0.04494
Balanced Accuracy      0.53120  0.49831  0.47531
```

table("SentLex_Sent" = df_unnested_2$comment_sentiment_lex, "class_manual"= df_unnested_2$class)

```
            class_manual
SentLex_Sent -1  0  1
          -3  1  1  0
          -2  4  7  1
          -1 29 74  9
          0   9 33  5
          1  13 58  9
          2   1 12  0
          4   0  1  0
```

table("SentLex_Sent" = df_unnested_2$comment_sentiment_lex1, "class_manual"= df_unnested_2$class)

```
            class_manual
SentLex_Sent  -1   0   1
          -1   5   8   1
          0   51 165  23
          1    1  13   0
```

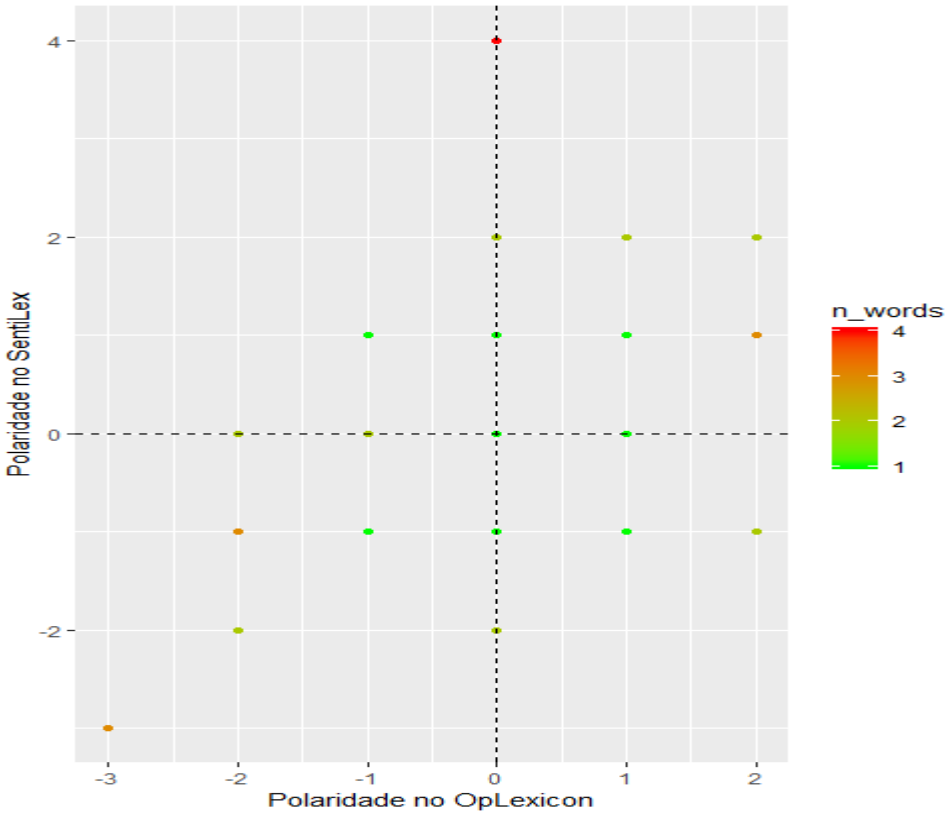
conf.mat <- confusionMatrix(df_unnested_2$comment_sentiment_lex1, df_unnested_2$class)

conf.mat

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   5   8   1
```

```
        0   51 165  23
        1    1  13   0
Overall Statistics
               Accuracy : 0.6367
                 95% CI : (0.5759, 0.6945)
    No Information Rate : 0.6966
    P-Value [Acc > NIR] : 0.9849
                  Kappa : -0.0077
 Mcnemar's Test P-Value : 1.872e-07
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity            0.08772  0.88710  0.00000
Specificity            0.95714  0.08642  0.94239
Pos Pred Value         0.35714  0.69038  0.00000
Neg Pred Value         0.79447  0.25000  0.90514
Prevalence             0.21348  0.69663  0.08989
Detection Rate         0.01873  0.61798  0.00000
Detection Prevalence   0.05243  0.89513  0.05243
Balanced Accuracy      0.52243  0.48676  0.47119
```

```
p <- df_unnested_2 %>%
   ggplot(aes(x = comment_sentiment_op, y = comment_sentiment_lex)) +
   geom_point(aes(color = n_words)) +
   scale_color_continuous(low = "green", high = "red") +
   labs(x = "Polaridade no OpLexicon", y = "Polaridade no SentiLex") +
   #geom_smooth(method = "lm") +
   geom_vline(xintercept = 0, linetype = "dashed") +
   geom_hline(yintercept = 0, linetype = "dashed")

p
```



most_pos <- which.max(df_unnested_2$most_pos)

most_neg <- which.min(df_unnested_2$most_neg)

cat(df$sentence[df$id == df_unnested$id[most_pos]])

cat(df$sentence[df$id == df_unnested$id[most_neg]])

usaremos o léxico OpLexicon para a análise de sentimento:

df_unnested_3 <- df_unnested %>% inner_join(op30, by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_op = sum(polarity), n_words = n()) %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_op),most_pos = max(comment_sentiment_op))

df_unnested_4 <- df_unnested %>% inner_join(sent %>% dplyr::select(term, lex_polarity = polarity), by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_lex = sum(lex_polarity), n_words = n()) %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_lex),most_pos = max(comment_sentiment_lex))

library(sqldf)

dfx <- sqldf("select * from df_unnested_3 left outer join df_unnested_4 on df_unnested_3.id = df_unnested_4.id UNION select * from df_unnested_4 left outer join df_unnested_3 on df_unnested_3.id = df_unnested_4.id")

```
dfxx <- sqldf("select id, class, comment_sentiment_op, comment_sentiment_lex from dfx group by id, class, comment_sentiment_op, comment_sentiment_lex ")

dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lex)] <- dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lex)]


dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lop)] <- dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lop)]


dfxx$comment_sentiment_op1 <- as.factor(ifelse(dfxx$comment_sentiment_op > 1, 1, ifelse(dfxx$comment_sentiment_op < -1, -1, 0)))


dfxx$comment_sentiment_lex1 <- as.factor(ifelse(dfxx$comment_sentiment_lex > 1, 1, ifelse(dfxx$comment_sentiment_lex < -1, -1, 0)))


dfyy <- sqldf("select id, class, comment_sentiment_op1, comment_sentiment_lex1 from dfxx group by id, class, comment_sentiment_op1, comment_sentiment_lex1")


nrow(dfyy)

# [1] 965  # 50 tweets não classificados


table("opLex_Sent" = dfyy$comment_sentiment_op1, " SentLex_Sent "= dfyy$comment_sentiment_lex1)

            SentLex_Sent
opLex_Sent  -1   0   1
        -1  52  45   0
        0   45 691  40
        1    0  40  52


dfyy$sent_LexOp <- as.factor(ifelse(dfyy$comment_sentiment_op1 == "1" & dfyy$comment_sentiment_lex1 == "1", 1, ifelse(dfyy$comment_sentiment_op1 == "-1" & dfyy$comment_sentiment_lex1 == "-1", -1, 0)))

dfzz <- sqldf("select id, class, sent_LexOp from dfyy group by id, class, sent_LexOp")

nrow(dfzz)

# [1] 880 # Existem 135 tweets que não foram classificados no modelo


df111 <- df11 %>% dplyr::select(id, class)

dfxxyy <- sqldf("select df111.id, df111.class from df111 left join dfzz on df111.id = dfzz.id where dfzz.id is null")

table(dfxxyy$class) # classificação dos tweets excluídos pelo modelo two-step


 -1   0   1
  7 120   8
```

Resumo Excluídos

	-1	0	1
Total	158	777	80
Excluidos	7	120	8
%_Exclud	4,43%	15,44%	10,00%

```
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
glimpse(df)
df$class <- as.factor(df$Opiniao)
df$id <- seq.int(nrow(df))
dff <- df %<>% inner_join(dfzz %>% dplyr::select(id, sent_LexOp), by = "id")
nrow(dff)
# [1] 880  # ref = 1.015

df_x1 <- subset(dff, sent_LexOp == 1)
nrow(df_x1)
# [1] 52
df_x0 <- subset(dff, sent_LexOp == -1)
nrow(df_x0)
# [1] 52
df_x00 <- subset(dff, sent_LexOp == 0)
nrow(df_x00)
# [1] 776


dff$sent_lex <- ifelse(dff$sent_LexOp ==1, 1, ifelse(dff$sent_LexOp == -1, -1, 0))

table("Lex_Sent" = dff$sent_lex, "class_manual"= dff$class)
        class_manual
Lex_Sent  -1   0   1
      -1   9  40   3
      0  128 588  60
      1    4  43   5

dff$sent_lex <- as.factor(dff$sent_lex)
conf.mat <- confusionMatrix(dff$sent_lex, dff$class)
```

conf.mat

```
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1   9  40   3
        0  128 588  60
        1    4  43   5

Overall Statistics

               Accuracy : 0.6841
                 95% CI : (0.6522, 0.7147)
    No Information Rate : 0.7625
    P-Value [Acc > NIR] : 1
                  Kappa : -0.0074
 Mcnemar's Test P-Value : 1.277e-10

Statistics by Class:

                     Class: -1 Class: 0 Class: 1
Sensitivity            0.06383   0.8763 0.073529
Specificity            0.94181   0.1005 0.942118
Pos Pred Value         0.17308   0.7577 0.096154
Neg Pred Value         0.84058   0.2019 0.923913
Prevalence             0.16023   0.7625 0.077273
Detection Rate         0.01023   0.6682 0.005682
Detection Prevalence   0.05909   0.8818 0.059091
Balanced Accuracy      0.50282   0.4884 0.507824
```

ANEXO 6 - CÓDIGO "R" PARA GERAÇÃO DE MODELOS "LEXINB"

```
setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library("data.table")
library(Rfacebook)
library(tidyverse)
library(tm)
library(ggExtra)
library(magrittr)
library(lubridate)
library(stringr)
library(tidytext)
library(lexiconPT)
library(caret)
library(e1071)

am_class <- read_excel("amostra_class1.xlsx", sheet="amostra_class") # Ver página 32
am_tema <- am_class %>%  select(sentence, Opiniao)

data("oplexicon_v3.0") # dataset de léxicos
data("sentiLex_lem_PT02") # dataset de léxicos
op30 <- oplexicon_v3.0
sent <- sentiLex_lem_PT02

set.seed(1)
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
df$class <- as.factor(df$Opiniao)
df$id <- seq.int(nrow(df))
df %>% mutate(id = row_number())

df_unnested <- df %>% unnest_tokens(term, sentence)
df_unnested_3 <- df_unnested %>% inner_join(op30, by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_op = sum(polarity), n_words = n()) %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_op),most_pos = max(comment_sentiment_op))
```

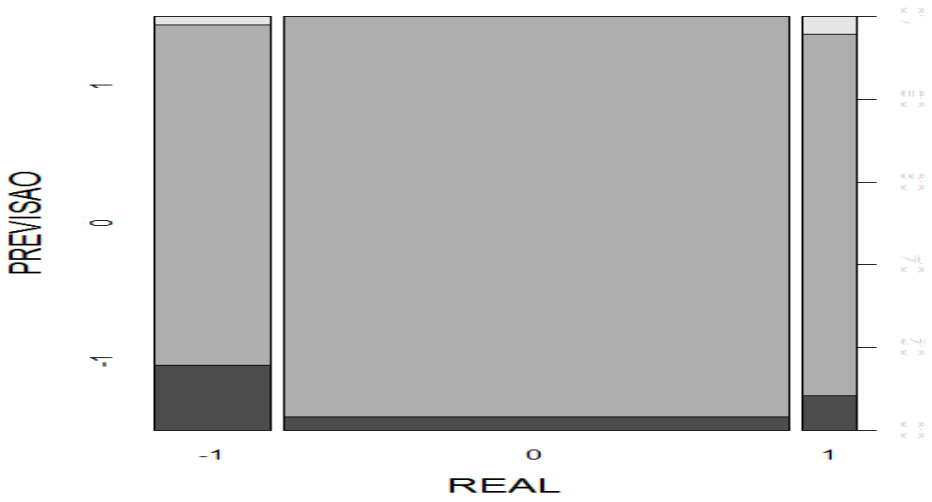
```
df_unnested_4 <- df_unnested %>% inner_join(sent %>% dplyr::select(term, lex_polarity = polarity), by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_lex = sum(lex_polarity), n_words = n())  %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_lex),most_pos = max(comment_sentiment_lex))


library(sqldf)


dfx <- sqldf("select * from df_unnested_3 left outer join df_unnested_4 on df_unnested_3.id = df_unnested_4.id UNION select * from df_unnested_4 left outer join df_unnested_3 on df_unnested_3.id = df_unnested_4.id")


dfxx <- sqldf("select id, class, comment_sentiment_op, comment_sentiment_lex from dfx group by id, class, comment_sentiment_op, comment_sentiment_lex ")


dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lex)] <- dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lex)]


dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lop)] <- dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lop)]


dfxx$comment_sentiment_op1 <- as.factor(ifelse(dfxx$comment_sentiment_op > 1, 1, ifelse(dfxx$comment_sentiment_op < -1, -1, 0)))


dfxx$comment_sentiment_lex1 <- as.factor(ifelse(dfxx$comment_sentiment_lex > 1, 1, ifelse(dfxx$comment_sentiment_lex < -1, -1, 0)))


dfyy <- sqldf("select id, class, comment_sentiment_op1, comment_sentiment_lex1 from dfxx group by id, class, comment_sentiment_op1, comment_sentiment_lex1")


dfyy$sent_LexOp <- as.factor(ifelse(dfyy$comment_sentiment_op1 == "1" & dfyy$comment_sentiment_lex1 == "1", 1, ifelse(dfyy$comment_sentiment_op1 == "-1" & dfyy$comment_sentiment_lex1 == "-1", -1, 0)))


dfyy$sent_LexOp <- as.factor(ifelse(dfyy$comment_sentiment_op1 == "1"
| dfyy$comment_sentiment_lex1 == "1", 1,
ifelse(dfyy$comment_sentiment_op1 == "-1" |
dfyy$comment_sentiment_lex1 == "-1", -1, 0)))


dfzz <- sqldf("select id, class, sent_LexOp from dfyy group by id, class, sent_LexOp")


dff <- df %<>% inner_join(dfzz %>% dplyr::select(id, sent_LexOp), by = "id")


set.seed(2)

dff1 <- dff[sample(1:nrow(dff)), ]

dff1 <- dff[sample(1:nrow(dff)), ]
```

```
stp <- c("o", "de", "a", "que", "e", "em", "do", "para", "os", "ao", "da", "com", "uma", "por", "na", "as", "um", "se", "como", "dos", "das", "no", "mas", "me")
corpus <- Corpus(VectorSource(dff1$sentence))
corpus.clean <- corpus %>%
  tm_map(content_transformer(tolower)) %>%
  tm_map(removePunctuation) %>%
  tm_map(removeNumbers) %>%
  tm_map(removeWords, stopwords(kind="pt")) %>%
  tm_map(removeWords, stp) %>%
  tm_map(stripWhitespace)
dtm1 <- DocumentTermMatrix(corpus.clean)

features <- findFreqTerms(dtm1, 3)
dtm2 <- DocumentTermMatrix(corpus.clean, list(global = c(2, Inf),dictionary = features))

train_idx <- createDataPartition(dff1$class, p=0.66, list=FALSE)
train1 <- dff1[train_idx,]
test1 <- dff1[-train_idx,]
train2 <- corpus.clean[train_idx]
test2 <- corpus.clean[-train_idx]

dict2 <- findFreqTerms(dtm2, lowfreq=3)
sms_train <- DocumentTermMatrix(train2, list(dictionary=dict2))
sms_test <- DocumentTermMatrix(test2, list(dictionary=dict2))

convert_count <- function(x) {
  y <- ifelse(x > 0, 1,0)
  y <- factor(y, levels=c(0,1), labels=c("No", "Yes"))
  y
}

trainNB1 <- apply(sms_train, 2, convert_count)
testNB1 <- apply(sms_test, 2, convert_count)
classifier <- naive*Bayes*(trainNB1, train1$class, laplace = 1)
pred <- predict(classifier, newdata=testNB1)
predt <- predict(classifier, newdata=trainNB1)
dff.test <- cbind(test1, pred)
dff.train <- cbind(train1, predt)
colnames(dff.train)[6] <- "pred"
dff_t <- rbind(dff.test, dff.train)
```

```
nrow( dff_t)

confmat_nb <- confusionMatrix(dff.test$pred,  dff.test$class)
confmat_nb
```

```
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1   8   7   2
        0   42 216  21
        1    1   0   1

Overall Statistics

               Accuracy : 0.755
                 95% CI : (0.7021, 0.8028)
    No Information Rate : 0.7483
    P-Value [Acc > NIR] : 0.4248

                  Kappa : 0.1526
 Mcnemar's Test P-Value : 4.817e-10

Statistics by Class:

                     Class: -1 Class: 0 Class: 1
Sensitivity            0.15686   0.9686 0.041667
Specificity            0.96356   0.1600 0.996350
Pos Pred Value         0.47059   0.7742 0.500000
Neg Pred Value         0.84698   0.6316 0.922297
Prevalence             0.17114   0.7483 0.080537
Detection Rate         0.02685   0.7248 0.003356
Detection Prevalence   0.05705   0.9362 0.006711
Balanced Accuracy      0.56021   0.5643 0.519009
```

```
tt1 <- data.frame(dff.test$class, dff.test$pred)
par(cex.lab=1.5)
plot(tt1, xlab= "REAL", ylab="PREVISÃO", cex.lab=2)
axis(4, col = "white", col.axis = "white", lwd = 0)
```

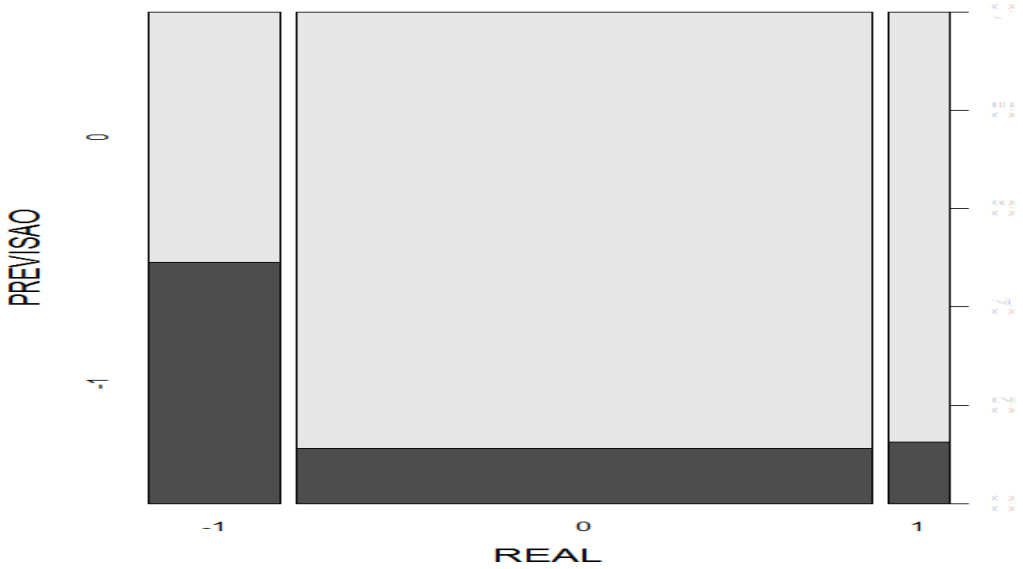



```
dff.test$prev_f <- as.factor(ifelse(dff.test$pred == "1" &
dff.test$sent_Lex == "1", 1, ifelse(dff.test$pred == "-1" |
dff.test$sent_Lex == "-1", -1, 0)))
confmat_nb <- confusionMatrix(dff.test$prev_f,  dff.test$class)
confmat_nb
```

```
Confusion Matrix and Statistics

          Reference
Prediction  -1   0   1
        -1  25  25   3
        0   26 198  21
        1    0   0   0

Overall Statistics

               Accuracy : 0.7483
                 95% CI : (0.695, 0.7966)
    No Information Rate : 0.7483
    P-Value [Acc > NIR] : 0.531

                  Kappa : 0.2897
 Mcnemar's Test P-Value : 2.475e-05

Statistics by Class:

                     Class: -1 Class: 0 Class: 1
Sensitivity            0.49020   0.8879  0.00000
Specificity            0.88664   0.3733  1.00000
Pos Pred Value         0.47170   0.8082      NaN
```

```
Neg Pred Value          0.89388   0.5283  0.91946
Prevalence              0.17114   0.7483  0.08054
Detection Rate          0.08389   0.6644  0.00000
Detection Prevalence    0.17785   0.8221  0.00000
Balanced Accuracy       0.68842   0.6306  0.50000
```

```
tt1 <- data.frame(dff.test$class, dff.test$prev_f)
par(cex.lab=1.5)
plot(tt1, xlab= "REAL", ylab="PREVISÃO", cex.lab=2)
axis(4, col = "white", col.axis = "white", lwd = 0)
```



ANEXO 7 - CÓDIGO "R" PARA GERAÇÃO DE MODELOS "SVM"

```
setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library("data.table")
library(tidyverse)
library(stringr)
library(tidytext)
library(tm)
library(textmineR)
library(RTextTools)
library(caret)
library(e1071)
library(pander)

am_class <- read_excel("amostra_class1.xlsx", sheet="amostra_class") # Ver página 32
am_tema <- am_class %>%  select(sentence, Opiniao)
am_tema$class <- as.factor(am_tema$Opiniao)
am_tema <- am_tema %>%  select(sentence, class)
nrow(am_tema)
[1] 1015

# Randomize the dataset
set.seed(1)
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
df$id <- seq.int(nrow(df))
dfx1 <- df %>%  select(id, sentence, class)

corpus <- Corpus(VectorSource(dfx1$sentence))
stopwords <- stopwords(kind = "pt")
stp <- c("o", "de", "a", "que", "e", "em", "do", "para", "os", "ao", "da", "com", "uma", "por", "na", "as", "um", "se", "como", "dos", "das", "no", "mas", "me")
corpus <- tm_map(corpus, removeWords, stopwords())
corpus <- tm_map(corpus, removeWords, stp)
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removePunctuation)
dtm <- DocumentTermMatrix(corpus)
```

```
wdtm <- weightTfIdf(dtm) # testado mas produziu resultafos inferiores

features <- findFreqTerms(dtm, 3)
dtm2 <- DocumentTermMatrix(corpus, list(global = c(2, Inf),dictionary = features))
inspect(dtm2)

train_idx <- createDataPartition(dfx1$class, p=0.66, list=FALSE)
train1 <- dfx1[train_idx,]
test1 <- dfx1[-train_idx,]
train2 <- corpus[train_idx]
test2 <- corpus[-train_idx]
```

```
frqtab <- function(x, caption) {round(100*prop.table(table(x)), 1)}
ft_orig <- frqtab(dfx1$class)
ft_train <- frqtab(train1$class)
ft_test <- frqtab(test1$class)
ft_df <- as.data.frame(cbind(ft_orig, ft_train, ft_test))
pander(head(ft_df), caption="Negativo/Neutro/Positivo in Test and Training Sets")

----------------------------------------
     ft_orig   ft_train   ft_test
-------- --------- ---------- ---------
 **-1**    15.6       15.6      15.4

 **0**     76.6       76.5      76.7

 **1**      7.9       7.9        7.8
----------------------------------------
Table: Ham/Spam in Test and Training Sets
```

```
dict2 <- findFreqTerms(dtm2, lowfreq=3)
sms_train <- DocumentTermMatrix(train2, list(dictionary=dict2))
sms_test <- DocumentTermMatrix(test2, list(dictionary=dict2))

convert_counts <- function(x) {x <- ifelse(x > 0, 1, 0)}
sms_train <- sms_train %>% apply(MARGIN=2, FUN=convert_counts)
sms_test <- sms_test %>% apply(MARGIN=2, FUN=convert_counts)

sms_train <- as.data.frame(sms_train)
sms_test <- as.data.frame(sms_test)
sms_train1 <- cbind(cat=factor(train1$class), sms_train)
sms_test1 <- cbind(cat=factor(test1$class), sms_test)
sms_train1<-as.data.frame(sms_train1)
```

```
sms_test1<-as.data.frame(sms_test1)
       # fit1 <- svm(cat~., data=sms_train1)
fit1 <- svm(cat~., data=sms_train1, scale = FALSE)
fit1.pred <- predict(fit1, na.omit(sms_test1))
fit1.perf <- table(na.omit(sms_test1$cat), fit1.pred, dnn=c("Actual", "Predicted"))
fit1.perf
      Predicted
Actual  -1   0   1
    -1   0  53   0
    0    0 264   0
    1    0  27   0

confmat_svm <- confusionMatrix(fit1.pred,  sms_test1$cat)
confmat_svm
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   0   0   0
        0   53 264  27
        1    0   0   0
Overall Statistics
               Accuracy : 0.7674
                 95% CI : (0.7191, 0.8111)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.5299
                  Kappa : 0
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity             0.0000   1.0000  0.00000
Specificity             1.0000   0.0000  1.00000
Pos Pred Value             NaN   0.7674      NaN
Neg Pred Value          0.8459      NaN  0.92151
Prevalence              0.1541   0.7674  0.07849
Detection Rate          0.0000   0.7674  0.00000
Detection Prevalence    0.0000   1.0000  0.00000
Balanced Accuracy       0.5000   0.5000  0.50000
```

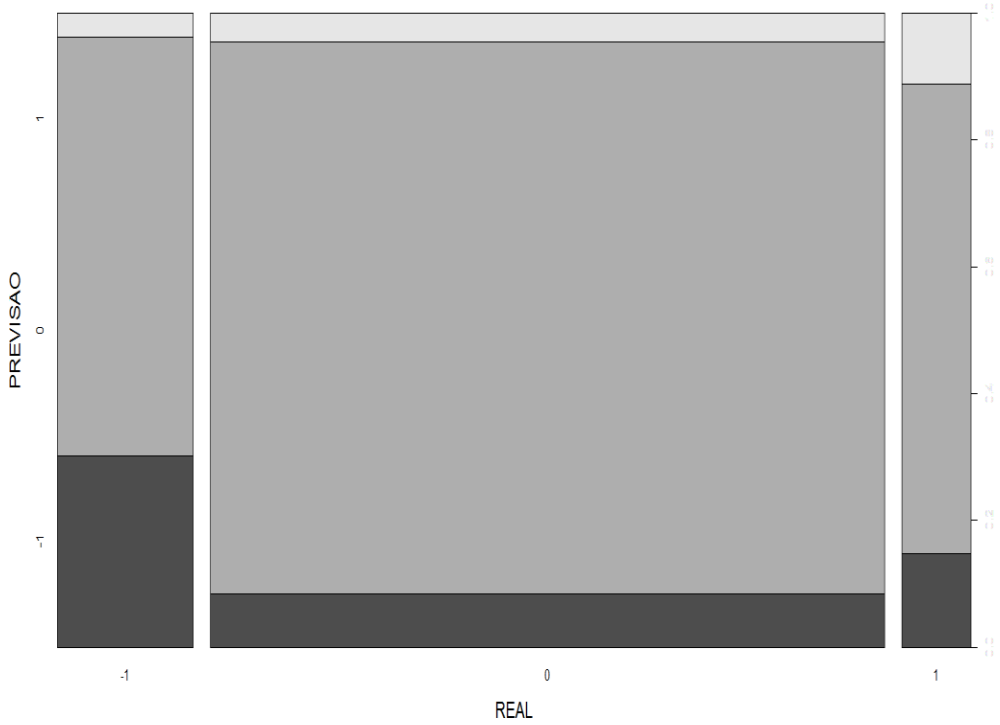
#Escolha do melhor modelo para kernel linear

```
fitLIN = svm(cat~., data=sms_train1, kernel = "linear", cost = 1, scale = FALSE)
fitLIN_pred <- predict(fitLIN, na.omit(sms_test1))
confmat_svm <- confusionMatrix(fitLIN_pred,  sms_test1$cat)
```

confmat_svm

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1  16  19   7
        0   33 237  13
        1    4   8   7

Overall Statistics
               Accuracy : 0.7558
                 95% CI : (0.7069, 0.8003)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.7197
                  Kappa : 0.2932
 Mcnemar's Test P-Value : 0.1229
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity            0.30189   0.8977  0.25926
Specificity            0.91065   0.4250  0.96215
Pos Pred Value         0.38095   0.8375  0.36842
Neg Pred Value         0.87748   0.5574  0.93846
Prevalence             0.15407   0.7674  0.07849
Detection Rate         0.04651   0.6890  0.02035
Detection Prevalence   0.12209   0.8227  0.05523
Balanced Accuracy      0.60627   0.6614  0.61070
```

```
tt1 <- data.frame(sms_test1$cat, fitLIN_pred)
par(cex.lab=1.5)
plot(tt1, xlab= "REAL", ylab="PREVISÃO", cex.lab=2)
axis(4, col = "white", col.axis = "white", lwd = 0)
```



tuneLIN <- tune(svm, cat~., data=sms_train1, kernel = "linear", ranges = list(cost = c(0.001, 0.01, 0.1, 1, 5, 10, 20, 100)))

(bestmod <- tuneLIN$best.model)

```
Call:

best.tune(method = svm, train.x = cat ~ ., data = sms_train1, ranges = list(cost =
c(0.001, 0.01, 0.1, 1, 5, 10, 20, 100)),

    kernel = "linear")

Parameters:

   SVM-Type:  C-classification

 SVM-Kernel:  linear

       cost:  0.001

      gamma:  0.0007358352

Number of Support Vectors:  438
```

svmtunLIN = svm(cat~., data=sms_train1, kernel = "linear", cost = 0.001,gamma = 0.0007358352 , scale = FALSE)

fit3.pred <- predict(svmtunLIN, na.omit(sms_test1))

confmat_svm <- confusionMatrix(fit3.pred, sms_test1$cat)

confmat_svm

```
Confusion Matrix and Statistics

          Reference

Prediction  -1   0   1

        -1   0   0   0
```

```
        0   53 264  27
        1    0   0   0
Overall Statistics
               Accuracy : 0.7674
                 95% CI : (0.7191, 0.8111)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.5299
                  Kappa : 0
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity             0.0000   1.0000  0.00000
Specificity             1.0000   0.0000  1.00000
Pos Pred Value             NaN   0.7674      NaN
Neg Pred Value          0.8459      NaN  0.92151
Prevalence              0.1541   0.7674  0.07849
Detection Rate          0.0000   0.7674  0.00000
Detection Prevalence    0.0000   1.0000  0.00000
Balanced Accuracy       0.5000   0.5000  0.50000
```

#Escolha do melhor modelo para kernel radial e polinomial

```
fitRAD <- svm(cat~., data=sms_train1, method="C-classification", kernal="radial",  gamma=0.1, cost=10, scale=FALSE)
fit4.pred <- predict(fitRAD, na.omit(sms_test1))
confmat_svm <- confusionMatrix(fit4.pred,  sms_test1$cat)
confmat_svm
```

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   6   6   0
        0   47 258  26
        1    0   0   1
Overall Statistics
               Accuracy : 0.7703
                 95% CI : (0.7222, 0.8138)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.4791
                  Kappa : 0.1028
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity            0.11321   0.9773 0.037037
Specificity            0.97938   0.0875 1.000000
```

```
Pos Pred Value         0.50000   0.7795 1.000000
Neg Pred Value         0.85843   0.5385 0.924198
Prevalence             0.15407   0.7674 0.078488
Detection Rate         0.01744   0.7500 0.002907
Detection Prevalence   0.03488   0.9622 0.002907
Balanced Accuracy      0.54629   0.5324 0.518519
```

tunerad <- tune(svm, cat~., data=sms_train1, kernel = "radial", ranges = list(cost = c(0.001,0.01,0.1,1,10,100,1000), gamma = c(0.1,0.5,1,2,3,4)), scale=FALSE)

fit5 <- tunerad$best.model

fit5.pred <- predict(fit5, na.omit(sms_test1))

fit5

```
Call:
best.tune(method = svm, train.x = cat ~ ., data = sms_train1, ranges = list(cost = 
c(0.001, 0.01, 0.1, 1, 10, 100, 1000),
    gamma = c(0.1, 0.5, 1, 2, 3, 4)), kernel = "radial", scale = FALSE)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  1
      gamma:  0.5
Number of Support Vectors:  659
```

confmat_svm <- confusionMatrix(fit5.pred, sms_test1$cat)

confmat_svm

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   0   1   0
        0   53 263  26
        1    0   0   1
Overall Statistics
               Accuracy : 0.7674
                 95% CI : (0.7191, 0.8111)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.5299
                  Kappa : 0.016
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity           0.000000   0.9962 0.037037
Specificity           0.996564   0.0125 1.000000
Pos Pred Value        0.000000   0.7690 1.000000
Neg Pred Value        0.845481   0.5000 0.924198
Prevalence            0.154070   0.7674 0.078488
```

```
Detection Rate         0.000000   0.7645 0.002907
Detection Prevalence   0.002907   0.9942 0.002907
Balanced Accuracy      0.498282   0.5044 0.518519
```

svmPOL <- svm(cat~., data=sms_train1, type='C-classification', kernel='polynomial', degree=8, gamma=0.1, coef0=1, scale=FALSE)

fit4.pred <- predict(svmPOL, na.omit(sms_test1))

confmat_svm <- confusionMatrix(fit4.pred, sms_test1$cat)

confmat_svm

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   3   3   0
        0   49 261  24
        1    1   0   3
Overall Statistics
               Accuracy : 0.7762
                 95% CI : (0.7284, 0.8191)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.3788
                  Kappa : 0.1092
 Mcnemar's Test P-Value : 3.567e-14
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity           0.056604   0.9886 0.111111
Specificity           0.989691   0.0875 0.996845
Pos Pred Value        0.500000   0.7814 0.750000
Neg Pred Value        0.852071   0.7000 0.929412
Prevalence            0.154070   0.7674 0.078488
Detection Rate        0.008721   0.7587 0.008721
Detection Prevalence  0.017442   0.9709 0.011628
Balanced Accuracy     0.523147   0.5381 0.553978
```

tunePOL <- tune(svm, cat~., data=sms_train1, kernel = "polynomial", ranges = list(cost = c(0.001,0.01,0.1,1,10,100,1000), gamma = c(0.1,0.5,1,2,3,4)), scale=FALSE)

fit6 <- tunePOL$best.model

fit6.pred <- predict(fit6, na.omit(sms_test1))

fit6

```
Call:
best.tune(method = svm, train.x = cat ~ ., data = sms_train1, ranges = list(cost =
c(0.001, 0.01, 0.1, 1, 10, 100, 1000),
    gamma = c(0.1, 0.5, 1, 2, 3, 4)), kernel = "polynomial", scale = FALSE)
Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  polynomial
```

```
       cost:  10
     degree:  3
      gamma:  0.1
     coef.0:  0
Number of Support Vectors:  666
```

```
fit6.pred <- predict(fit6, na.omit(sms_test1))
confmat_svm <- confusionMatrix(fit6.pred,  sms_test1$cat)
confmat_svm
```

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   2   2   0
        0   51 262  26
        1    0   0   1
Overall Statistics
               Accuracy : 0.7703
                 95% CI : (0.7222, 0.8138)
    No Information Rate : 0.7674
    P-Value [Acc > NIR] : 0.4791
                  Kappa : 0.0498
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity           0.037736   0.9924 0.037037
Specificity           0.993127   0.0375 1.000000
Pos Pred Value        0.500000   0.7729 1.000000
Neg Pred Value        0.850000   0.6000 0.924198
Prevalence            0.154070   0.7674 0.078488
Detection Rate        0.005814   0.7616 0.002907
Detection Prevalence  0.011628   0.9855 0.002907
Balanced Accuracy     0.515431   0.5150 0.518519
```

ANEXO 8 - **CÓDIGO "R" PARA GERAÇÃO DE MODELOS "LEXISVM"**

```
setwd("C:/…/5_TESE/AnaliseSentimentos")
library("readxl")
library(tm)
library("data.table")
library(Rfacebook)
library(tidyverse)
library(ggExtra)
library(magrittr)
library(lubridate)
library(stringr)
library(tidytext)
library(lexiconPT)
library(caret)
library(e1071)
library(sqldf)
library(pander)

am_class <- read_excel("amostra_class1.xlsx", sheet="amostra_class") # Ver página 32
am_tema <- am_class %>%  select(sentence, Opiniao)
data("oplexicon_v3.0") # dataset de léxicos
data("sentiLex_lem_PT02") # dataset de léxicos
op30 <- oplexicon_v3.0
sent <- sentiLex_lem_PT02
set.seed(1)
df <- am_tema[sample(1:nrow(am_tema)), ]
df <- am_tema[sample(1:nrow(am_tema)), ]
df$class <- as.factor(df$Opiniao)
df$id <- seq.int(nrow(df))
df_unnested <- df %>% unnest_tokens(term, sentence)

df_unnested_3 <- df_unnested %>% inner_join(op30, by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_op = sum(polarity), n_words = n()) %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_op),most_pos = max(comment_sentiment_op))

df_unnested_4 <- df_unnested %>% inner_join(sent %>% dplyr::select(term, lex_polarity = polarity), by = "term") %>% group_by(id, class) %>% summarise(comment_sentiment_lex =
```

```
sum(lex_polarity), n_words = n())  %>% ungroup() %>% rowwise() %>% mutate(most_neg = min(comment_sentiment_lex),most_pos = max(comment_sentiment_lex))


dfx <- sqldf("select * from df_unnested_3 left outer join df_unnested_4 on df_unnested_3.id = df_unnested_4.id UNION select * from df_unnested_4 left outer join df_unnested_3 on df_unnested_3.id = df_unnested_4.id")


dfxx <- sqldf("select id, class, comment_sentiment_op, comment_sentiment_lex from dfx group by id, class, comment_sentiment_op, comment_sentiment_lex ")


dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lex)] <- dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lex)]


dfxx$comment_sentiment_op[is.na(dfxx$comment_sentiment_lop)] <- dfxx$comment_sentiment_lex[is.na(dfxx$comment_sentiment_lop)]


dfxx$comment_sentiment_op1 <- as.factor(ifelse(dfxx$comment_sentiment_op > 1, 1, ifelse(dfxx$comment_sentiment_op < -1, -1, 0)))


dfxx$comment_sentiment_lex1 <- as.factor(ifelse(dfxx$comment_sentiment_lex > 1, 1, ifelse(dfxx$comment_sentiment_lex < -1, -1, 0)))


dfyy <- sqldf("select id, class, comment_sentiment_op1, comment_sentiment_lex1 from dfxx group by id, class, comment_sentiment_op1, comment_sentiment_lex1")


dfyy$sent_LexOp <- as.factor(ifelse(dfyy$comment_sentiment_op1 == "1" & dfyy$comment_sentiment_lex1 == "1", 1, ifelse(dfyy$comment_sentiment_op1 == "-1" & dfyy$comment_sentiment_lex1 == "-1", -1, 0)))


dfzz <- sqldf("select id, class, sent_LexOp from dfyy group by id, class, sent_LexOp")


df <- am_tema[sample(1:nrow(am_tema)), ]

df <- am_tema[sample(1:nrow(am_tema)), ]


df$class <- as.factor(df$Opiniao)

df$id <- seq.int(nrow(df))

dff <- df %<>% inner_join(dfzz %>% dplyr::select(id, sent_LexOp), by = "id")


dff$sent_lex <- ifelse(dff$sent_LexOp ==1, 1, ifelse(dff$sent_LexOp == -1, -1, 0))

table("Lex_Sent" = dff$sent_lex, "class_manual"= dff$class)

dff$sent_lex <- as.factor(dff$sent_lex)


stp <- c("o", "de", "a", "que", "e", "em", "do", "para", "os", "ao", "da", "com", "uma", "por", "na", "as", "um", "se", "como", "dos", "das", "no", "mas", "me")
```

```
corpus <- Corpus(VectorSource(dff$sentence))
stopwords <- stopwords(kind = "pt")
corpus <- tm_map(corpus, removeWords, stopwords())
corpus <- tm_map(corpus, removeWords, stp())
corpus <- tm_map(corpus, stripWhitespace)
corpus <- tm_map(corpus, removePunctuation)
dtm <- DocumentTermMatrix(corpus)
features <- findFreqTerms(dtm, 3)
dtm2 <- DocumentTermMatrix(corpus, list(global = c(2, Inf),dictionary = features))
train_idx <- createDataPartition(dff$class, p=0.66, list=FALSE)
train1 <- dff[train_idx,]
test1 <- dff[-train_idx,]
train2 <- corpus[train_idx]
test2 <- corpus[-train_idx]

dict2 <- findFreqTerms(dtm2, lowfreq=3)
sms_train <- DocumentTermMatrix(train2, list(dictionary=dict2))
sms_test <- DocumentTermMatrix(test2, list(dictionary=dict2))

convert_counts <- function(x) {x <- ifelse(x > 0, 1, 0)}
sms_train <- sms_train %>% apply(MARGIN=2, FUN=convert_counts)
sms_test <- sms_test %>% apply(MARGIN=2, FUN=convert_counts)

sms_train <- as.data.frame(sms_train)
sms_test <- as.data.frame(sms_test)
sms_train1 <- cbind(cat=factor(train1$class), sms_train)
sms_test1 <- cbind(cat=factor(test1$class), sms_test)
sms_train1<-as.data.frame(sms_train1)
sms_test1<-as.data.frame(sms_test1)
classifier <- svm(cat~., data=sms_train1, scale = FALSE)
pred <- predict(classifier, newdata= na.omit(sms_test1))
predt <- predict(classifier, newdata= na.omit(sms_train1))
dff.test <- cbind(test1, pred)

confmat_nb <- confusionMatrix(dff.test$pred,  dff.test$class)
confmat_nb
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
```

```
        -1   0   0   0
        0   45 229  24
        1    0   0   0
Overall Statistics
               Accuracy : 0.7685
                 95% CI : (0.7163, 0.8151)
    No Information Rate : 0.7685
    P-Value [Acc > NIR] : 0.5322
                  Kappa : 0
 Mcnemar's Test P-Value : NA
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
Sensitivity              0.000   1.0000  0.00000
Specificity              1.000   0.0000  1.00000
Pos Pred Value             NaN   0.7685      NaN
Neg Pred Value           0.849      NaN  0.91946
Prevalence               0.151   0.7685  0.08054
Detection Rate           0.000   0.7685  0.00000
Detection Prevalence     0.000   1.0000  0.00000
Balanced Accuracy        0.500   0.5000  0.50000
```

```
tt1 <- data.frame(dff.test$class, dff.test$pred)
par(cex.lab=1.5)
plot(tt1, xlab= "REAL", ylab="PREVISÃO", cex.lab=2)
axis(4, col = "white", col.axis = "white", lwd = 0)
```

```
dff.test$prev_f <- as.factor(ifelse(dff.test$pred == "1" | dff.test$sent_Lex == "1", 1,
ifelse(dff.test$pred == "-1" & dff.test$sent_Lex == "-1", -1, 0)))
confmat_nb <- confusionMatrix(dff.test$prev_f,  dff.test$class)
confmat_nb
```

```
Confusion Matrix and Statistics
          Reference
Prediction  -1   0   1
        -1   0   0   0
        0   42 212  20
        1    3  17   4
Overall Statistics
               Accuracy : 0.7248
                 95% CI : (0.6704, 0.7748)
    No Information Rate : 0.7685
    P-Value [Acc > NIR] : 0.9662
                  Kappa : 0.041
 Mcnemar's Test P-Value : 8.214e-10
Statistics by Class:
                     Class: -1 Class: 0 Class: 1
```

Sensitivity	0.000	0.9258	0.16667
Specificity	1.000	0.1014	0.92701
Pos Pred Value	NaN	0.7737	0.16667
Neg Pred Value	0.849	0.2917	0.92701
Prevalence	0.151	0.7685	0.08054
Detection Rate	0.000	0.7114	0.01342
Detection Prevalence	0.000	0.9195	0.08054
Balanced Accuracy	0.500	0.5136	0.54684