|  |
| --- |
| brasao |
| Silvano Nogueira Buback  Utilizando aprendizado de máquina para construção de uma ferramenta de apoio a moderação de comentários |
|  |
| Dissertação de mestrado  Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio.  Orientador: Prof. Marco Antonio Casanova |
|  |
| Rio de Janeiro, 01 de setembro de 2011. |

|  |
| --- |
| brasao |
| Silvano Nogueira Buback  Utilizando aprendizado de máquina para construção de uma ferramenta de apoio a moderação de comentários |
|  |
| Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre pelo Programa de Pós-graduação em Informática do Departamento de Informática do Centro Técnico e Científico da PUC-Rio. Aprovada pela Comissão Examinadora abaixo assinada. |
| Prof. Marco Antonio Casanova  Orientador  PUC-Rio  Prof. Rui L. Milidiú  Co-Orientador  PUC-Rio  Prof. Antonio L. Furtado  PUC-Rio  Prof. Karin K. Breitman  PUC-Rio  **Prof. José Eugenio Leal**  Coordenador Setorial do Centro  Técnico Científico – PUC-Rio  Rio de Janeiro, 01 de setembro de 2011. |

|  |  |
| --- | --- |
|  | Todos os direitos reservados. É proibida a reprodução total ou parcial do trabalho sem autorização da universidade, do autor e do orientador. |
|  | Silvano Nogueira Buback  Graduou-se em Ciência da Computação com ênfase em Sistemas de Informação na Universidade Federal do Espírito Santo onde ajudou a construir um ambiente de desenvolvimento de software baseado em ontologias. Trabalhou na Americanas.com e na Globo.com onde é responsável pela engenharia dos sistemas de autenticação e cadastro de usuários e de comentários. Desenvolveu como dissertação de mestrado uma ferramenta para auxiliar a moderação de comentários de notícias. |
|  | Ficha Catalográfica |
|  | Silvano Nogueira Buback  Utilizando aprendizado de máquina para construção de uma ferramenta de apoio a moderação de comentários / Silvano Nogueira Buback; orientador: Marco Antonio Casanova. - Rio de Janeiro: PUC-Rio, Departamento de Informática, 2011.  v., 64 f.: il. ; 29,7 cm  Dissertação de Mestrado - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática  Referencias bibliográficas incluídas.  Moderação de comentários; Aprendizado de Máquina; SVM; Classificadores Bayesianos; Boosting; Processamento de Linguagem Natural; Classificação de texto |

|  |
| --- |
| Agradecimentos |
| Agradeço primeiramente à Deus, que me conduziu e me protegeu durante toda essa caminhada.  À minha esposa Karol, pela imensa compreensão e paciência durante esses anos de mestrado, inclusive o apoio na revisão da dissertação.  À Globo.com, por ter acreditado em mim e oferecido a oportunidade de cursar este mestrado.  Ao meus colegas da Globo.com, pelas idéias e pela força que me deram para a conclusão desta dissertação.  Ao meu orientador Marco Antonio Casanova pela paciência, compreensão e pelos ensinamentos.  Ao meu co-orientador Ruy Milidiú pelas aulas e pelo conhecimento compartilhados.  Aos meus pais pela compreensão de eu não estar tão presente, e mesmo assim eles estão sempre torcendo por mim. |

|  |
| --- |
| Resumo |
| Silvano Nogueira Buback. **Utilizando aprendizado de máquina para construção de uma ferramenta de apoio a moderação de comentários.** Rio de Janeiro, 2011. 64p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.  Uma das mudanças trazidas pela Web 2.0 é a maior participação dos usuários na produção do conteúdo, através de opiniões em redes sociais ou comentários nos próprios sites de produtos e serviços. Estes comentários são muito valiosos para seus sites pois fornecem feedback e incentivam a participação e divulgação do conteúdo. Porém excessos podem ocorrer através de comentários com palavrões indesejados ou spam. Enquanto para alguns sites a própria moderação da comunidade é suficiente, para outros as mensagens indesejadas podem comprometer o serviço. Para auxiliar na moderação dos comentários foi construída uma ferramenta que utiliza técnicas de aprendizado de máquina para auxiliar o moderador. Para testar os resultados, dois corpora de comentários produzidos na Globo.com foram utilizados, o primeiro com 657.405 comentários postados diretamente no site, e outro com 451.209 mensagens capturadas do Twitter. Nossos experimentos mostraram que o melhor resultado é obtido quando se separa o aprendizado dos comentários de acordo com o tema sobre o qual esta sendo comentado. |
| Palavras-chave  Moderação de comentários; Aprendizado de Máquina; SVM; Classificadores Bayesianos; Boosting; Processamento de Linguagem Natural; Classificação de texto. |

|  |
| --- |
| Abstract |
| Silvano Nogueira Buback. **Using Machine Learning to build a tool that helps comments moderation.** Rio de Janeiro, 2011. 64p. Dissertação de Mestrado - Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.  One of the main changes brought by Web 2.0 is the increase of user participation in content generation mainly in social networks and comments in news and service sites. These comments are valuable to the sites because they bring feedback and motivate other people to participate and to spread the content. On the other hand these comments also bring some kind of abuse as bad words and spam. While for some sites their own community moderation is enough, for others this impropriate content may compromise its content. In order to help theses sites, a tool that uses machine learning techniques was built to mediate comments. As a test to compare results, two datasets captured from Globo.com were used : the first one with 657.405 comments posted through its site and the second with 451.209 messages captured from Twitter. Our experiments show that best result is achieved when comment learning is done according to the subject that is being commented. |
| Keywords  Comments Moderation; Machine Learning; SVM; Bayesian Classifiers; Boosting; Natural Language Processing; Text Classification. |

|  |
| --- |
| Sumário |
|  |

1 Introdução 11

1.1. Organização da tese 12

2 Trabalhos relacionados 13

2.1. Filtro anti-spam 13

2.2. Sentiment Analysis 14

3 Modelagem do sistema 17

3.1. Técnicas para aprendizado adotadas 17

3.1.1. Naïve Bayes 17

3.1.2. Boostexter 18

3.1.3. SVM 19

3.2. Métodos de extração de atributos 20

3.2.1. Bag of Words 20

3.2.2. Corretor 21

3.2.3. POS Tagging 22

3.2.4. N-grams 23

3.3. Arquitetura 23

3.4. Visão geral da arquitetura 24

3.4.1. Camada de Aprendizagem 26

4 Experimentos 30

4.1. Critérios para avaliação do sistema 30

4.1.1. Precisão 30

4.1.2. Recall 31

4.1.3. Acurácia 31

4.1.4. F-measure 31

4.1.5. Coeficiente de correlação 32

4.2. Corpora 32

4.2.1. Corpus 1 33

4.2.1. Corpus 2 33

4.2.2. Diferenças entre os corpus 33

4.3. Metodologia do teste 34

4.4. Resultados obtidos 35

4.4.1. Primeira etapa 36

4.4.2. Segunda etapa 37

4.4.3. Terceira etapa 39

4.5. Discussão 42

5 Conclusões e trabalhos futuros 45

6 Bibliografia 47

Apêndice A – Resultados da etapa I dos experimentos 50

Apêndice B – Distribuição dos comentários por grupo 61

Apêndice C – Parâmetros de uso da API de linha de comando 63

|  |
| --- |
| Lista de imagens |
|  |

Figura 1 – Relacionamento entre as entidades do sistema de comentários 24

Figura 2 - Arquitetura do sistema 25

Figura 3 - Diagrama de classes da camada aprendizagem 27

Figura 4 - Validação cruzada para 4 partições 35

Figura 5 - Gráfico da probabilidade de rejeição por percentagem de erro 40

Figura 6 - Gráfico da quantidade de comentários por probabilidade de rejeição 41

Figura 7 - Gráfico da contribuição do erro geral por probabilidade 41

|  |
| --- |
| Lista de tabelas |
| Tabela 1 - Resultados obtidos em (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000) e (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000). 13  Tabela 2 - Configuração das máquinas utilizadas nos testes 36  Tabela 3 - Resultados obtidos utilizando classificadores de contexto 39  Tabela 4 - Desvio padrão variando-se o parâmetro C (SVM) 42  Tabela 5 - Desvio padrão variando-se o método de seleção de atributos 43 |

# Introdução

Com 2 bilhões de usuários (MINIWATTS MARKETING GROUP, 2011) e mais de 120 milhões de websites (DOMAINTOOLS.COM, 2011), a Internet já alcança 28,7% da população mundial. Para atrair cada vez mais usuários, dentre outras estratégias, os sites estão se tornando mais participativos. Isto pode ser conseguido, principalmente, com integração às redes sociais, incentivo à contribuição de conteúdo, fóruns, listas de discussão ou comentários. Porém com a enorme quantidade de usuários interagindo, torna-se muito difícil controlar o conteúdo gerado. É provável ocorrer alguns excessos, como o uso indevido para disseminação de spams, vírus, pornografia e outras mensagens indesejadas.

Para alguns sites, a própria comunidade de usuários faz a moderação, através de funcionalidades como denuncie e mecanismos de reputação, dentre outros. O problema desta solução é que ela é pós-moderada, ou seja, a mensagem é sempre exibida e, de acordo com a classificação dos usuários, ela poderá ser removida. Porém, até que isto ocorra, vários usuários já a leram.

Entretanto, existem sites em que uma única mensagem indesejada, mesmo que entre muitas, não é sequer admissível. Imagine um comentário pornográfico um site construído para crianças. Certamente, os mecanismos de pós-moderação não serviriam para este caso. Para alguns sites é necessário que a mensagem seja pré-moderada, ou seja, antes que qualquer usuário do site possa vê-la, um moderador, autorizado pelo site, vai validá-la e aprovar sua publicação, caso o conteúdo seja compatível. Porém, para sites com grande volume de acessos, a quantidade de moderadores necessários para fazer toda a moderação pode inviabilizar o negócio ou muitas contribuições serão deixadas de lado, desmotivando os usuários de participar. Para resolver este problema, o objetivo da dissertação será construir um sistema de moderação de comentários automático, capaz de aprender novos padrões automaticamente através de um moderador humano e auxiliá-lo na tarefa de moderação.

A dificuldade para a realização deste trabalho está na identificação do critério do que deve ser aprovado ou reprovado, pois normalmente ele é subjetivo. Às vezes é necessário um entendimento do contexto para avaliar se um determinado comentário poderá ser aprovado ou não. Por exemplo, o comentário feito através do Twitter na página da novela Passione durante a campanha eleitoral:

Lá vem o Serra no comercial de #passione

Este comentário foi bloqueado apesar de não possuir nenhuma palavra obscena, pois se tratava de um comentário sobre política e certamente a emissora de TV não iria permitir comentários políticos na página da novela, especialmente durante a campanha, pois precisa manter a imparcialidade.

Apesar da palavra “Serra" ser um indicativo de que o comentário devesse ser reprovado no exemplo acima, o mesmo não deveria ocorrer se o comentário aparecesse em uma página de debate político. Ou seja, é necessário ainda que o sistema leve em consideração o contexto que o comentário aparece para ser mais preciso na classificação. Porém, não é objetivo do trabalho verificar se o comentário pertence ao contexto em que foi inserido, somente ser especializado ao contexto no momento da moderação.

## Organização da tese

Esse trabalho está organizado da seguinte forma. O Capítulo 2 descreve os trabalhos relacionados a este. O Capítulo 3 apresenta a modelagem adotada, incluindo a parte referente aos algoritmos para auto-moderação. O Capítulo 4 expõe alguns experimentos feitos com dois corpora reais e os resultados obtidos. O Capítulo 6 resume os resultados obtidos e sugere trabalhos futuros.

# Trabalhos relacionados

Não foram encontrados trabalhos diretamente relacionados a auto-moderação de comentários. Contudo este trabalho possui uma semelhança com trabalhos que tratam de filtros anti-spam. Como muitos deles estão ligados a linguagem natural, utilizando técnicas de aprendizado de maquina, também serão estudados trabalhos sobre análise sentimental. Estes trabalhos servirão como guia para o desenvolvimento da dissertação e também ajudarão a construir um base-line.

## Filtro anti-spam

Dentro os artigos pesquisados, (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000) e (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000) são muito citados. Nestes trabalhos, os autores mostram como utilizar Naive Bayes (LEWIS, 1998) para construção de um filtro anti-spam e qual o impacto que a seleção dos atributos possui no resultado final. Como métrica é utilizado uma acurácia com peso λ para mensagens legítimas que são classificadas incorretamente como spam.

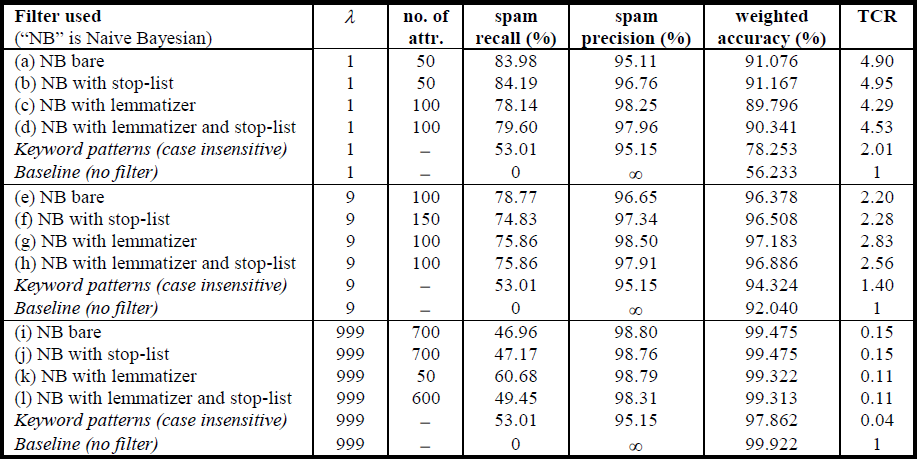


Tabela 1 - Resultados obtidos em (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000) e (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000).

A utilização de um peso para medir a acurácia é útil, pois os autores entendem que classificar como spam uma mensagem legítima é mais grave que classificar como mensagem legítima um spam. Este aspecto é muito importante para o trabalho de auto-moderação de comentários, porém o peso λ possui um efeito inverso. Neste caso é mais grave aprovar uma mensagem que deveria ser reprovada, pois tal ação pode ter um efeito muito negativo para a página que ela está sendo exibida. Se uma mensagem aprovada fosse reprovada, ainda poderíamos contar com a ajuda manual do moderador.

Conforme já publicado por (JOACHIMS, 1998), SVM tem-se mostrado eficiente em tarefas de classificação de texto em geral. (LAI, 2007) fez um estudo sobre a acurácia do Naive Bayes, k-NN e SVM para classificação de spam, utilizando dois corpora. Em ambos, o SVM mostrou-se mais eficiente que os outros algoritmos, chegando atingir acurácia de 94,59%, contra 81,59% do k-NN e 90,89% do Naive Bayes. Este resultados são os melhores resultados de cada algoritmo utilizando o corpus 2, com 30% do corpus para teste. (DRUCKER, WU e VAPNIK, 1999) também publicaram um estudo sobre o SVM e também concluíram que ele possui uma maior acurácia para a classificação de spam.

Para tentar melhorar ainda mais a acurácia, alguns trabalhos utilizam técnicas para combinar algoritmos. (CARRERAS e MÀRQUEZ, 2001) implementaram uma variante do AdaBoost e conseguiram precisão de 98,73% com recall de 97,09%, para λ=1. (SAKKIS, ANDROUTSOPOULOS, *et al.*, 2001) combinaram vários algoritmos com stacked generalization e conseguiram melhorar a acurácia de (ANDROUTSOPOULOS, KOUTSIAS, *et al.*, 2000), especialmente quando λ>1.

## Sentiment Analysis

Analisando trabalhos sobre classificação de texto, encontramos um tema de muito interesse atualmente: *Sentiment Analysis*. Este consiste em uma classificação de texto em que se está interessado em saber se o mesmo tem um sentimento positivo ou negativo (alguns trabalhos consideram também neutro). Este tema ganhou muita evidência devido à proliferação das redes sociais. Ele permite, em larga escala, saber qual o sentimento dos usuários sobre um contexto definido, que pode ser um produto, pessoa, ou serviço.

O trabalho mais conhecido sobre sentiment analysis é (PANG, LEE e VAITHYANATHAN, 2002). Este trabalho parece ser o primeiro a utilizar técnicas de aprendizado de máquina para resolver o problema. Nele os autores mostram que contar palavras, selecionadas manualmente, que intuitivamente fornecem uma identificação sobre o sentimento (como brilhante, excelente ou terrível), não dão uma boa acurácia. A partir daí eles utilizam 3 algoritmos de aprendizado de máquina, Máxima Entropia (BERGER, PIETRA e PIETRA, 1996), Naïve Bayes (LEWIS, 1998) e SVM (JOACHIMS, 1998), combinando técnicas diferentes para a seleção dos atributos. Os autores mostram que o SVM utilizando como atributo somente a presença ou não das palavras e símbolos de pontuação tem a melhor acurácia. A análise também é feita com Part of Speech (POS), somente de adjetivos e adicionado as palavras sua posição no texto. Os autores ainda argumentam que sentiment analysis parece ser uma tarefa mais desafiadora do que a tradicional classificação de texto em tópicos.

Mais recentemente, (ALVIM, VILELA, *et al.*, 2010) propuseram o uso do SVM com Bigrams e POS Tagging conseguindo acurácia de 84,80% sob o mesmo corpus de (PANG, LEE e VAITHYANATHAN, 2002), com 3 folds, e 86,09% com 10 folds, tornando-se o estado da arte para este corpus. Utilizando a mesma técnica num corpus de notícias da Petrobrás (em português), (ALVIM, VILELA, *et al.*, 2010) conseguiram acurácia de 84,00%, desta vez com 5 folds; porém, para este corpus, a acurácia foi melhor quando se utilizou como atributo apenas a presença das palavras com unigrams, com acurácia de 85,94%.

Outro trabalho proposto sobre sentiment analysis é (TURNEY, 2002). Uma característica deste trabalho que difere bastante dos demais é que ele utiliza um método não-supervisionado, ou seja, não é necessário um corpus para fazer o aprendizado. Alternativamente, os autores utilizam um algoritmo baseado em três etapas, a saber:

1. Identificar os adjetivos e advérbios através de um algoritmo de POS Tag.
2. Calcular a orientação semântica de cada frase através de um algoritmo de PMI-IR (CHURCH e HANKS, 1990), porém fazendo acesso a um *search engine* para obter informações sobre cada palavra selecionada na primeira etapa e o quão mais aproximada de *excellent* ou *poor* ela estaria. A isto era atribuído um valor numérico.
3. Por fim, atribuir uma classe - recomendado ou não-recomendado - de acordo com a média da orientação semântica das frases calculadas no item acima.

Observe que este trabalho mostra uma heurística que não depende de um corpus para fazer o aprendizado (não-supervisionado), porém precisa de um acesso a um serviço Web de busca para avaliar se o adjetivo ou advérbio em questão define uma boa avaliação ou não, o que poderia tornar o algoritmo inviável. Além disto, a acurácia obtida não é melhor que o método proposto por (PANG, LEE e VAITHYANATHAN, 2002), sendo de apenas 65,83% para filmes, porém a comparação não é feita com o mesmo corpus.

# Modelagem do sistema

Neste capítulo será apresentado as técnicas para aprendizagem adotadas e como o sistema foi modelado para suportá-las.

## Técnicas para aprendizado adotadas

Pesquisando na literatura, encontramos muitos algoritmos para aprendizado supervisionado. Cada um utiliza uma técnica diferente para determinar a função aprendizado e adéqua-se melhor de acordo com o tipo de problema.

Embora a técnica utilizada não seja a mesma em todos os algoritmos citados aqui, todos precisam que o objeto a ser classificado, no caso o comentário, seja representado através de um vetor multidimensional chamado de *feature vector*, ou simplesmente *feature* ou, como iremos denominar, *atributo*. Na Seção 3.2 será detalhado os métodos de extração de atributos que utilizamos neste trabalho.

### Naïve Bayes

Naïve Bayes é um classificador probabilístico baseado no Teorema de Bayes, que mostra como determinar a probabilidade de um evento condicional através da probabilidade inversa. Para facilitar a computação, este classificador assume que a presença (ou ausência) de um atributo não tem relação alguma com qualquer outro atributo (por isto o nome Naive).

Partindo do teorema de Bayes:

Como é assumido que os atributos são independentes:

Em vez de calcular explicitamente, o algoritmo calcula apenas o denominador para cada classe e normaliza-os para que a soma seja 1. A este termo damos o nome de evidência.

Se estivermos interessado apenas em qual classe tem maior probabilidade, a evidência pode ser ignorada pois é uma constante, determinada durante o treino.

Para a classificação binária sabemos que a probabilidade das classes, para os mesmos atributos, são complementares, assim precisamos apenas ter a probabilidade do comentário ser rejeitado.

Quando a probabilidade de rejeição for menor que 0.5 o comentário é classificado como aprovado. Quando for maior que 0.5 é classificado como rejeitado. Quando a probabilidade é igual a 0.5, definimos que ele será rejeitado.

Para este trabalho estamos utilizando a implementação do classificador Naïve Bayes do toolkit NLTK (BIRD e LOPER, 2011). Embora esta biblioteca não tenha a implementação mais performática, ela oferece uma boa documentação e é implementada em Python, que é a mesma linguagem no qual este trabalho será implementado.

### Boostexter

O Boostexter é uma implementação de um algoritmo de classificação de texto baseado da técnica de *boosting*, em que diversos classificadores mais simples e de baixa acurácia são combinados em um único classificador com alta acurácia. Mais especificamente, para o *Adaboost*, os classificadores são treinados sequencialmente, sendo que cada classificador subsequente é treinado em exemplos que foram mais difíceis de classificar pelos classificadores anteriores (SCHAPIRE e SINGER, 2000).

O Boostexter pode ser implementado tanto sobre o Adaboost.MH quanto o Adaboost.MR, ambos projetados para classificação de dados que podem ser classificados simultaneamente em várias classes. A principal diferença entre o Adaboost.MH e o Adaboost.MR está no fato do primeiro trabalhar minimizando a perda de *Hamming* enquanto o segundo foi projetado para encontrar uma hipótese que ranqueia as classes de maneira a garantir que as corretas sempre estejam no topo da lista (SCHAPIRE e SINGER, 2000).

Muitas implementações de Adaboost utilizam redes neurais e árvore de decisão como classificadores internos (que serão combinados). Na implementação para classificação de texto é utilizado como classificador interno *stumps* (árvores de decisão de um único nível), em que a raiz avalia a presença ou ausência de um atributo no texto.

Para teste trabalho foi utilizado uma implementação de código aberto do Boostexter chamada Icsiboost (FAVRE e HAKKANI, 2011).

### SVM

A teoria sobre o Support Vector Machine (SVM) foi proposta inicialmente por (VAPNIK e LERNER, 1963), como um método eficiente para a identificação de padrões. Ela possui muitas vantagens na resolução de problemas com uma amostra pequena, não-lineares e com um grande número de dimensões. (JOACHIMS, 1998) foi o primeiro a introduzi-lo para a classificação textual.

Ela é principalmente baseada nas seguintes considerações (YONG-FENG e YAN-PING, 2004):

* Minimizar o risco estrutural, minimizando a dimensão VC (Vapnik-Chervonenkis) de um conjunto mínimo de funções para controlar o risco estrutural do aprendizado de máquina, de modo a torna-lo eficiente.
* Maximizar a distância entre classes (encontrar o hiperplano de classificação ótimo) para realizar o controle da dimensão VC, que pode ser garantido por um teorema da estatística.
* Utilizar funções kernel para trabalhar num espaço de maior dimensão, porém linear.

Comparado com os tradicionais métodos de classificação, o SVM tem a vantagem de evitar overfitting e oferece boa performance para classificação de texto, o que também depende do tipo de kernel escolhido. Aliás o critério de escolha do tipo de kernel e seus parâmetros são um ponto fraco do algoritmo, pois ainda são muito baseados em experiências humanas.

Para este trabalho foi escolhido a implementação do SVM de (FAN, CHANG, *et al.*, 2008), chamada de *LIBLINEAR* (nome devido a função kernel utilizada ser linear). Ela é uma versão otimizada especialmente para problemas de classificação de documento onde o número de atributos é maior que o número de exemplos de treino.

Uma outra limitação do SVM é o fato de apenas suportar atributos numéricos. Assim, para converter a representação de texto em vetores, a cada palavra do texto é atribuído um índice numérico sequencial e para cada coordenada do vetor é atribuído o valor 1 se a palavra está presente na instância, ou 0 caso contrário. Como o número de palavras em uma instância é sempre muito menor que o total de palavras existentes, a matriz que representa a presença ou ausência de uma palavra no texto tende a ser esparsa, não sendo necessário representar os atributos onde ela é zero para cada instância.

## Métodos de extração de atributos

Para que os classificadores citados acima consigam determinar uma função de aprendizado é necessário que os comentários sejam convertidos em vetores, selecionados de maneira a fornecer o máximo de informação sobre a probabilidade de rejeição do comentário. Estes vetores são chamados de *feature vectors*, ou simplesmente *features* ou atributos.

A escolha de uma função responsável por converter os comentários em vetores é fundamental para o bom desempenho de quaisquer um dos algoritmos utilizados neste trabalho, pois ela que irá conter as partes do comentário que irá relacioná-lo com a probabilidade de ser reprovado ou aprovado.

### Bag of Words

Um dos métodos de extração de atributos mais simples e mais utilizado em problemas de classificação de texto é o saco de palavras (*bag of words*). Neste método, cada palavra é um atributo do texto. Neste tipo de extração somente palavras e números são utilizados como atributo, ignorando todos os sinais de pontuação. Uma variação deste método é quando os caracteres de pontuação também são incluídos.

Devido a grande quantidade de erros de ortografia presente no corpus, as palavras são convertidas para minúscula e todos os acentos são removidos. Isto evita que “não” e “nao” sejam considerados como atributos diferentes, porém faz “vovô” e “vovó” serem considerados como mesmo atributo.

### Corretor

Analisando os corpora utilizados neste trabalho, percebemos que muitas palavras possuem erros de ortografia. No caso do corpus 1 (comentários de notícias), são comuns erros devido a digitação. Já para o corpus 2 (Twitter) muitos erros de ortografia são parte do estilo de escrita deste tipo de rede social (como “naum” em vez de “não”). Certamente estes erros de ortografia têm impacto no classificador, pois geram atributos diferentes para a mesma palavra. Para evitar isto, construímos um método de extração de atributos que utilizasse sempre uma versão corrigida da palavra como atributo ou a própria, caso já estivesse correta de acordo com um dicionário.

Uma heurística foi então construída para a correção das palavras, baseado no estudo do corpus. Inicialmente a palavra é verificada em 2 dicionários: de português e de inglês. Caso ela esteja em algum deles está correta. Caso não esteja, os métodos abaixo são aplicados até que se consiga obter uma palavra ortograficamente correta.

1. Tenta-se retirar os caracteres repetidos das palavras um a um e para cada nova palavra gerada sua existência é confirmada nos dicionários. A verificação a cada palavra repetida é necessária para evitar que a palavra, ortograficamente errada, “carrro” após a correção vire a palavra “caro”.
2. Palavras terminadas com “aum” ganham o sufixo “ão” e novamente o dicionário é verificado.
3. Expressões comuns como “haha...”, “rsrs...” e “auau....” são alteradas para “ha”, “rs” ou “au” independente da quantidade de repetições existentes ou de iniciarem com a segunda letra e/ou terminarem com a primeira (como “ahahah”).
4. Se a palavra não existe no dicionário mais foi redigida com inicial maiúscula e todas as outras minúsculas é considerada um nome próprio e assumimos estar correta. Por exemplo, “Google” ou “Microsoft”.
5. Caso nenhuma das técnicas acima tenham corrigido a palavra então buscamos sugestões no dicionário de português. Os seguintes critério então são utilizados:
   1. Caso nenhuma sugestão seja fornecida, a própria palavra é retornada. Neste caso estamos assumindo que é uma nova palavra ortograficamente correta.
   2. Se existe uma palavra na sugestão que difere a palavra escrita apenas pelos caracteres de acentuação, esta será a correção
   3. Por fim, é assumido como grafia correta da palavra aquela que possui a menor distancia de edição, calculado de acordo com (LEVENSHTEIN, 1966).

Os dicionários de português e inglês utilizados foram da biblioteca Aspell, na versão 0.60.6. O algoritmo para geração de sugestão utilizado foi o provido pelo próprio dicionário, e que reporta uma acurácia de 89,7% para sugestões em inglês. Os resultados detalhados e o corpus utilizado estão publicados em (GNU ASPELL, 2011).

### POS Tagging

Outro método muito utilizado para geração de atributos para a classificação de texto é o uso de POS Tagging, onde distinguimos as palavras também pela sua classe gramatical. Por exemplo, veja os 2 comentários abaixo e observem o uso da palavra “porcos”:

(1) “*Se um cão produz isso, imagina a quantidade de animais, entre* ***porcos****, bois, galinhas...*” (aprovado)

(2) “*Ficam esperando esses shows porcos ai e ...*” (reprovado)

No segundo exemplo, o comentário foi reprovado devido a existência da palavra “porcos” como adjetivo. No primeiro exemplo, a palavra “porcos” é um substantivo e por isto não fornece um indicativo de reprovação.

Como o texto possui muitos erros de ortografia, que prejudicariam a eficiência do classificador gramatical, antes do comentário ser classificado gramaticalmente, suas palavras são corrigidas pelo corretor ortográfico.

Foram utilizadas 2 implementações de POS Tagging para este trabalho. Uma treinada com o classificador Naive Bayes e outra utilizando um classificador sequencial, baseado apenas em memorizar as ocorrências mais comuns, como: (art,“décimo”) 🡪 adj, ou seja, a palavra “décimo” após um artigo é classificada como adjetivo. Apesar das diferenças entre os algoritmos de classificação a acurácia de ambos é muito parecida, em torno de 87% para o corpus Mac Morpho (ALUÍSIO, PELIZZONI, *et al.*, 2003). O que os diferencia é o tempo de classificação, superior no classificador sequencial.

### N-grams

Este não é propriamente um método para a geração de atributos, mas sim uma forma de combinar os atributos gerados pelos outros métodos, agrupando-os dois-a-dois (bigrams) ou três-a-três (trigrams). Somente são combinados os atributos vizinhos. Para entender melhor veja o exemplo abaixo:

“*Assisti a matéria do Jornal Hoje do segundo dia...*”

Para este exemplo a combinação de 2 n-grams (bigrams) seria: “Assisti a”; “a matéria”; “matéria do”; “do Jornal”; “Jornal Hoje”; “Hoje do”; “do segundo”; “segundo dia”

Para evitar uma combinação muito grande entre atributos, o que aumentaria o tempo de classificação e poderia causar overfitting, limitamos o número de n-grams a tamanho 3 (trigrams). Ainda, iremos nos referir daqui a diante por bigrams sempre a combinação de unigrams + bigrams, o mesmo para trigrams que é uma combinação de unigrams + bigrams + trigrams. Unigrams é quando os atributos não são combinados entre si.

A vantagem de utilizar n-grams maior que 1 é permitir que nomes compostos possam ser identificados como um único atributo. No exemplo acima, “Jornal Hoje” seria considerado um único atributo.

## Arquitetura

Neste trabalho estamos interessados apenas em mostrar como pode ser construída uma camada de apoio a moderação para sistemas de comentários. Sendo assim, vamos tomar como base o sistema de comentários utilizado na Globo.com para explicar alguns conceitos sobre sistemas de comentários, sabendo que os conceitos básicos utilizados aqui permitem que este trabalho seja aplicado em qualquer outro sistema do tipo.

O sistema de comentários da Globo.com é responsável pelos comentários em diversos sites, tanto da própria Globo.com quanto da TV Globo. Para evitar que os comentários sobre economia sejam moderados por um moderador de esportes ou da novela, cada tópico (ou assunto) comentado é classificado em categorias de acordo com a área em que está inserido. São categorias existentes: g1-economia, g1-educação, globoesporte-flamengo, globoesporte-vôlei entre outros. Embora o propósito desta classificação tenha sido criado para controle de acesso, indiretamente ela ajuda a definir um tema, ou **contexto**, agrupando os comentários de assuntos comuns em uma mesma categoria. Em geral, os comentários dentro da mesma categoria possuem alguns critérios comuns para aprovação ou reprovação. Esta classificação dos assuntos, de acordo com o contexto, será útil ao final do trabalho para construção de classificadores específicos ao contexto.

Para entender conceitualmente a relação entre as entidades no sistema de comentários, veja o diagrama da . Nele podemos observar que temos diversos comentários sobre um Tópico e que este é classificado em uma categoria.



Figura – Relacionamento entre as entidades do sistema de comentários

## Visão geral da arquitetura

Com o intuito de facilitar a manutenção e evolução do ambiente e permitir a produção de componentes de reuso, o sistema de moderação foi dividido em camadas, cada um responsável por uma parte do sistema.

Para conseguir auxiliar o moderador na classificação, foram utilizados algoritmos de aprendizado de máquina, o que levou a construção de uma camada para este propósito, com o nome de *aprendizagem*. Esta camada, é ainda subdividida em duas: *classificadores*, onde está a implementação dos algoritmos de classificação de comentários e a camada *extratores*, onde está a implementação dos métodos de extração de atributos.



Figura 2 - Arquitetura do sistema

Com a camada de aprendizagem isolada, é possível reutilizá-la para fins de experimentação, com corpus e parâmetros variados, e também para integração com a ferramenta de comentários. Para experimentação uma API de linha de comando foi construída, permitindo combinar todos os extratores com os algoritmos de classificação. O apêndice C detalha o uso desta API. Para integração com o sistema de comentários, basta este enviar os comentários classificados manualmente para o treino do moderador e para cada comentário novo consultar a predição.

Na figura 2 o diagrama de classes da camada de aprendizagem é apresentado. Como este trabalho foi implementado em linguagem Python, algumas considerações serão feitas:

* Python não possui atributos privados, mas possui um mecanismo para evitar conflito de atributos entre classes na herança. Os atributos com esta característica devem iniciar com \_\_, mas não podem terminar com tal. Embora isto não seja estritamente um atributo privado, foi tratado aqui como tal, pois é o mecanismo mais próximo do conceito oferecido pela linguagem. Apesar da necessidade do atributo iniciar com duplo sublinhado, no diagrama de classes o duplo sublinhado não está presente pois é subentendido pela marcação de privado.
* Python não possui classes abstratas nem interface. Para indicar que uma classe é abstrata ou interface foi incluído no nome da classe o sufixo I. Observe que isto é apenas uma convenção deste trabalho.

Nas seções abaixo as camadas serão detalhas, mostrando seus componentes internos e as classes que a implementam.

### Camada de Aprendizagem

É a principal camada, responsável por obrigar os algoritmos de moderação, que inclui os classificadores e os métodos de extração de atributos. O diagrama da figura 3 mostra as classes existentes e a relação entre elas.



Figura 3 - Diagrama de classes da camada aprendizagem

A classe Comment é responsável por representar um comentário. Quando ele está classificado, o atributo status indica sua classe, que pode ser aprovado (*APPROVED*) ou rejeitado (*REJECTED*).

As classes responsáveis por realizar a extração dos atributos de um comentário herdam de *FeatureExtractorI* e devem implementar um método chamado extract responsável por retornar os atributos a partir de um comentário.

Abaixo segue a descrição de cada subclasse de FeatureExtractorI. Os detalhes sobre os métodos são descritos na seção .

* SimpleFE: implementa o método de extração descrito na seção , porém considerando como atributos somente palavras e atributos.
* WordsNumberPuctuationFE: análogo ao SimpleFE, mas também considera como atributo sinais de pontuação.
* CorrectorFE: Análogo ao SimpleFE. Porém corrige a grafia das palavras automaticamente, utilizando para isto o corretor ortográfico Aspell (ATKINSON, 2004). As regras para correção ortográfica estão descritas na seção
* PosTaggerFE: Este método utiliza as palavras corrigidas pelo CorrectorFE e adiciona a elas a classificação gramatical. O método utilizado para classificar gramaticalmente o texto está descrito na seção
* MostInformativeFE: Este *decorator* (GAMMA, HELM, *et al.*, 1994) é responsável por utilizar somente os atributos mais informativos.
* NGramsFE: Este *decorator* é responsável por combinar os atributos da classe decorada em subconjuntos de n-grams. A seção detalha o processo.

Já as classes responsáveis por realizar o aprendizado e a classificação de comentários têm como classe pai a classe ModeratorI, que implementa o padrão de projeto Template method (GAMMA, HELM, *et al.*, 1994). Os únicos métodos que precisam ser implementados pelas subclasses são \_\_init\_\_, que é o construtor e deverá realizar o treino do algoritmo e o método which\_probability\_to\_be\_rejected, que deve retornar um número entre 0 e 1.0 com a probabilidade de um comentário ser rejeitado. A probabilidade do comentário ser aprovado é complementar a rejeição.

A classe ModeratorI possui ainda um atributo que merece destaque: strict\_protection. Quando este atributo contém o valor verdadeiro, os comentários que possuem tags html, códigos javascripts, links ou e-mails são diretamente rejeitos, sem necessidade de consultar o algoritmo de moderação, porém eles ainda são utilizados para treino.

Abaixo a descrição das subclasses de ModeratorI. Os detalhes da implementação dos algoritmos são descritos no Capítulo 3.

* BoostingModerator: Utiliza o algoritmo Boostexter para moderação.
* SVMModerator: Utiliza o algoritmo SVM para moderação.
* NBModerator: Utiliza o algoritmo Naive Bayes para moderação.
* Baseline: Classifica os comentários de acordo com a classe que mais ocorre no conjunto de treino. Utilizado apenas para comparativo dos algoritmos e para testes.
* SpecificLearningModerator: Um *decorator* que utiliza os classificadores especificados acima para moderar, construindo modelos específicos a cada categoria do comentário e um modelo global. Os detalhes sobre o funcionamento deste moderador está descrito na Seção .

# Experimentos

Para avaliar a eficiência do sistema de classificação automática, foi necessário submeter os algoritmos e os métodos de extração de atributos citados no capítulo 3 a um experimento. Para isto foram utilizados dois corpora com mensagens reais, já classificadas por vários moderadores manualmente.

Esta seção está organizada da seguinte forma. A Seção 4.1 mostra vários critérios que poderiam ser utilizados para avaliar a eficiência do nosso sistema e apresenta qual será utilizado. A Seção 4.2 descreve os corpora utilizados. A seção 4.3 aborda a metodologia para realizar o experimento utilizando os corpora. Por fim, a Seção 4.4 apresenta os resultados obtidos.

## Critérios para avaliação do sistema

A escolha do algoritmo e do método de extração de atributos depende da escolha de um critério de desempenho. Os critérios que foram avaliados para este trabalho são: *precisão*, *recall*, *acurácia*, *F-measure* e *coeficiente de correlação*. Nas seções abaixo faremos uma discussão sobre os critérios citados e suas aplicações, utilizando as seguintes abreviações:

* RC significa a quantidade de comentários reprovados corretamente.
* AC significa a quantidade de comentários aprovados corretamente.
* RI significa a quantidade de comentários reprovados incorretamente, ou seja, comentários que foram reprovados pelo classificador mas que deveriam ter sido aprovados.
* AI significa a quantidade de comentários aprovados incorretamente, ou seja, comentários que foram aprovados pelo classificador mas deveriam ter sido reprovados.

### Precisão

Mede a quantidade de comentários aprovados que o sistema reprovou (reprovados incorretamente). É definida pela fórmula:

Para maximizar a precisão é necessário que o moderador automático não reprove incorretamente os comentários, ou seja, mensagens que deveriam ser aprovadas mas foram classificadas como reprovadas. A desvantagem desta métrica é que ela não leva em consideração os comentários que deveriam ter sido reprovados mas foram aprovados.

### Recall

Mede a quantidade de comentários reprovados que o sistema aprovou. É definido pela fórmula:

Se relacionarmos este trabalho à pesquisa de filtros anti-spam, o recall seria uma métrica mais adequada do que a precisão. A desvantagem deste tipo de métrica é que ela não leva em consideração todas as medidas, assim, se um moderador automático reprovasse todos os comentários independente do conteúdo, este teria o recall máximo, embora não fosse eficiente.

### Acurácia

Mede o quão efetivo o sistema é do ponto de vista da classificação -mensagem aprovada ou reprovada. É definida pela fórmula:

O problema desta métrica é que ela não reflete a realidade quando o corpus não possui a mesma quantidade de elementos por classe, que é o caso deste trabalho. De fato, imagine um corpus em que 90% dos comentários foram aprovados e 10% reprovados. Se fizéssemos um classificador que aprovasse todos os comentários, independente do conteúdo, ele iria possuir uma acurácia de 90%.

### F-measure

É uma média harmônica entre o recall e a precisão. É definida pela fórmula:

Ela mede a eficiência do sistema levando em consideração o erro nas duas classes. É necessário que o acerto nas duas classes aumente para que a métrica aumente. Também permite que sejam definidos pesos diferentes para as classes através do coeficiente . Nos casos em que é definido como 1, esta métrica recebe o nome de .

### Coeficiente de correlação

Também conhecido como *Matthews Correlation Coeficient* (MCC), é uma medida de qualidade de classificadores binários. Ele é um coeficiente de correlação entre o observado e o predito, sendo um valor entre -1 e 1, com 1 indicando uma perfeita predição, -1 uma predição invertida e 0 uma predição aleatória. É calculado através da fórmula:

Uma característica importante do MCC para este trabalho é que ele pode ser utilizado mesmo quando as classes possuem diferentes tamanhos. Ele também leva em consideração os erros e acertos em todas as classes, diferentemente da precisão ou do recall.

Tanto quanto o são boas métricas para avaliar qual o melhor algoritmo a ser utilizado. Porém, neste trabalho vamos utilizar mais o MCC pois além de levar em consideração todas as medidas (RC, AC, RI e AI), o valor numérico desta métrica possui uma interpretação, indicando o quão próximo da perfeição o algoritmo encontra-se. Portanto, quando no restante do texto falarmos sobre eficiência do classificador, estaremos levando em consideração o valor da métrica .

## Corpora

Para os experimentos, dois corpora serão utilizados, descritos nesta seção.

### Corpus 1

Este corpus consiste em comentários reais postados em diversos sites de notícias da Globo.com (incluindo sites de programas da TV Globo). Todos os comentários deste corpus passaram por moderação manual.

Possui um total de 657.405 comentários divididos em 105 categorias, onde 573.821 (87,29%) dos comentários foram aprovados e 83.584 (12,71%) foram reprovados. No anexo B encontra-se uma tabela com a distribuição dos comentários por categoria.

### Corpus 2

Consiste de mensagens capturadas através da rede social Twitter para participação popular em alguns eventos feito pela TV Globo. Possui 451.209 mensagens (ou neste caso tweets) divididas em 25 categorias. Do total de comentários, 348.355 (77%) foram reprovadas e 102.854 (33%) foram aprovadas. No anexo B encontra-se uma tabela com a distribuição dos comentários por categoria.

### Diferenças entre os corpus

Apesar dos dois corpora serem referentes ao mesmo propósito de classificação, eles possuem algumas características que distinguem um do outro e que merecem ser citadas:

* A proporção entre comentários aprovados e reprovados é praticamente inversa entre os 2 corpora, sendo que o primeiro corpus possuiu mais mensagens aprovadas, enquanto que o segundo a maior parte das mensagens foram rejeitadas. Isto provavelmente ocorre porque os comentários do corpus 2 foram selecionados através da busca de palavras (ou hashtags, que são marcações de palavras, mas que se iniciam com um caractere # e normalmente indicam um assunto do qual se fala, como #passione quando se fala da novela). Portanto, muitas das mensagens poderiam não interessar ao contexto na qual ela deveria ser inserida, sendo portanto rejeitada pelo moderador.

Ao contrário, o corpus 1 foi coletado através de comentários que foram postados na página em que se discutia um assunto. Portanto a moderação manual neste caso é referente a mensagens indevidas por não se tratarem do assunto ou por possuir termos abusivos. O objetivo do corpus 2 será de confrontar os resultados obtidos no experimento com o corpus 1.

* No corpus 1 as mensagens possuem mais palavras por comentário, em média, e menos erros de ortografia. Isto deve-se à limitação das mensagens na rede social Twitter, limitadas em 140 caracteres, fazendo que os textos sejam mais abreviados.

## Metodologia do teste

Como é necessário utilizar uma parte do corpus para treino e outra para testes, utilizamos um método conhecido como validação cruzada (*cross validation*). Neste método a amostra original é aleatorizada antes do teste e dividida em *k* partes com mesmo tamanho, onde*(k-1)* partes são utilizadas para treino e 1 parte é utilizada para testes. Este processo é repetido *k* vezes, alternando em cada uma delas a parte utilizada para testes, sem repetição, de modo que todos os conjuntos de mensagens escolhidas para treino também servirão como teste e vice-versa. A figura 4 ilustra melhor o processo:



Figura 4 - Validação cruzada para 4 partições

Para minimizar a oscilação na medida, visto que o corpus para treino e para teste é aleatorizado e torna-se diferente a cada teste, vamos utilizar 10 partições, minimizando a incerteza entre execuções subsequentes.

## Resultados obtidos

Nesta seção iremos mostrar os resultados obtidos. O experimento foi separado em 3 etapas diferentes. A primeira etapa determina qual algoritmo e método de seleção de atributos possui a melhor resposta para o problema. A segunda etapa, com o algoritmo e o método de extração escolhidos, avalia a eficiência adicionada ao sistema, levando-se em consideração o contexto. A última etapa analisa a distribuição das probabilidades do comentário ser reprovado e o erro de acordo com a probabilidade.

### Primeira etapa

Consiste na escolha do melhor método de extração/algoritmo a ser utilizado para este problema. Dentre as possibilidades testadas estão os algoritmos SVM, Naive Bayes e Boostexter. Uma tentativa de análise utilizando o Maximum Entropy foi feita, mas este algoritmo possui um tempo de classificação muito alto e a implementação escolhida consumia muita memória, o que inviabilizou o teste. Eles foram combinados com os métodos de extração (citados no Capítulo 3), simples (letras e números apenas), com pontuação, com correção de palavras e com classificação gramatical (utilizando NaiveBayes e também uma implementação mais simples e eficiente). Na combinação dos algoritmos e métodos de extração também aplicamos n-grams, limitados a tamanho 3.

Devido ao tamanho dos corpora e às implementações dos algoritmos acima necessitarem do corpus inteiro em memória, foram alugadas máquinas virtuais na Amazon do tipo High Memory Extra Large Instance (*m2.xlarge*). Abaixo uma descrição detalhada da configuração da máquina virtual:

Tabela 2 - Configuração das máquinas utilizadas nos testes

|  |  |
| --- | --- |
| Configuração | Valor |
| CPU | 6.5EC2 Compute Units |
| Cores | 2 (virtuais) |
| Arquitetura | 64 bits |
| RAM | 17.1 GB |
| Disco | 50 G |

Como o ambiente de execução é virtualizado, houve uma variação grande no tempo de processamento, razão pela qual esta métrica não foi incluída nos resultados, que se encontram no Anexo A.

Analisando as tabelas do Anexo A, podemos verificar que o SVM com tri-grams foi mais eficiente (em relação ao MCC) nos dois corpora. Porém, para o corpus 1, o uso de palavras e números como atributos trouxe um melhor resultado, enquanto que no corpus 2, ao adicionarmos os caracteres de pontuação (WNL), o MCC aumentou.

Como o corpus 2 é baseado em mensagens do Twitter, provavelmente o caractere “#” está auxiliando o classificador, pois nestas mensagens ele é um forte indicativo de contexto. Os outros caracteres como “:)” são um indicador de sentimento e também podem contribuir para auxiliar a moderação, especialmente devido ao fato do corpus 2 ter comentários para os Web sites da TV; neste contexto comentários que falam bem têm maior chance de serem aprovados. Como estes símbolos não ocorrem muito nos comentários do corpus 1, adicioná-los apenas gerou ruído para o classificador.

Conforme citado na Seção 4.2.2, o corpus 1 é o corpus de referência, enquanto que o corpus 2 foi utilizado para confrontar os resultados. Concluímos que o SVM com tri-grams e uso como atributos apenas palavras (sem sinais de pontuação) é a melhor combinação para o experimento na etapa II..

### Segunda etapa

O problema da abordagem da classificação da primeira etapa é que os atributos possuem igual peso, independentemente do contexto no qual estão inseridos. Assim, o moderador automático não consegue distinguir entre termos que poderiam ser utilizados num contexto, mas que não poderiam ser utilizados em outros. Para ilustrar melhor o problema, considere comentários sobre sexo num Web site jornalístico e em um Web site para adolescentes. Certamente, a maioria dos comentários com referencias sexuais não seriam permitidos neste último.

#### Implementação do moderador de contexto

Para conseguir atingir este objetivo, foi necessário construir um moderador que levasse em consideração o contexto no qual o comentário está inserido. Este contexto é identificado unicamente pelo código do grupo do comentário.

Uma primeira abordagem para a implementação deste moderador de contexto seria adicionar o grupo como atributo dos comentários. O problema desta abordagem é que os algoritmos de classificação de texto não conseguem relacionar todas as combinações de atributos. Por isto utilizar bi-grams ou tri-grams, em geral, aumenta a performance do algoritmo pois as palavras (ou atributos) lado-a-lado são relacionadas.

Sendo assim passamos para uma abordagem de criar um moderador para cada grupo. Porém esta solução ainda encontra limitações pois muitos grupos possuem poucos comentários e a quantidade de memória necessária para manter todos os classificadores seria proibitiva. Além disto, e mais importante, não haveria um aprendizado global ou compartilhado sobre atributos que deveriam sempre rejeitar um comentário.

Por fim, a solução adotada para o problema foi combinar até dois classificadores, como os utilizados no experimento I, por grupo. Um classificador global e, se necessário, um outro classificador específico do grupo. Os classificadores por grupo eram criados apenas quando o grupo possuía pelo menos 1.000 comentários para treino e estavam restrito a no máximo 25 moderadores, para os grupos com mais comentários de treino.

A implementação deste moderador de contexto é feita pela classe ContextModerador. Internamente, esta classe decide quais grupos iriam ter moderadores e durante o treino os comentários eram treinados com o moderador global e, caso houvesse, com o moderador do grupo. Quando um comentário é enviado para a classificação o moderador de contexto, obtém a probabilidade do moderador global e também do moderador específico caso exista. A probabilidade de rejeição utilizada é do classificador que tem maior certeza da sua classificação. Em termos matemático, seja o comentário que está sendo classificado, a probabilidade do comentário pelo moderador global e a probabilidade para o moderador específico (do grupo).

#### Resultados

A tabela 3 mostra os resultados obtidos quando se levou em consideração o contexto dos comentários na moderação. Como o classificador que melhor se ajustou ao problema de moderação foi o SVM com tri-grams e utilizando-se somente letras e números como atributos, obtido na etapa I, utilizamos ele também para o classificador global e dos grupos.

Tabela 3 - Resultados obtidos utilizando classificadores de contexto

| Corpus | RC | AC | RI | AI | Acurácia | Recall | **F1** | MCC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Corpus 1 | 54,424 | 83,264 | 574,141 | 657,405 | 89,10% | 34,64% | 0.446 | 0.412 |
| Corpus 2 | 314,428 | 62,476 | 40,378 | 33,927 | 83,53% | 90,26% | 0.894 | 0.522 |

Utilizando classificadores especializados por grupo, todas as métricas de performance aumentaram. Analisando o corpus 1 podemos verificar que o MCC saiu de 0.375 para 0.412, um aumento de quase 10%. A métrica F1 também aumentou de 0.423 para 0.446. Mesmo a acurácia, que não estamos considerando plenamente pois o corpus não é balanceado, foi para 89,10%, maior que qualquer outro classificador utilizando no corpus 1.

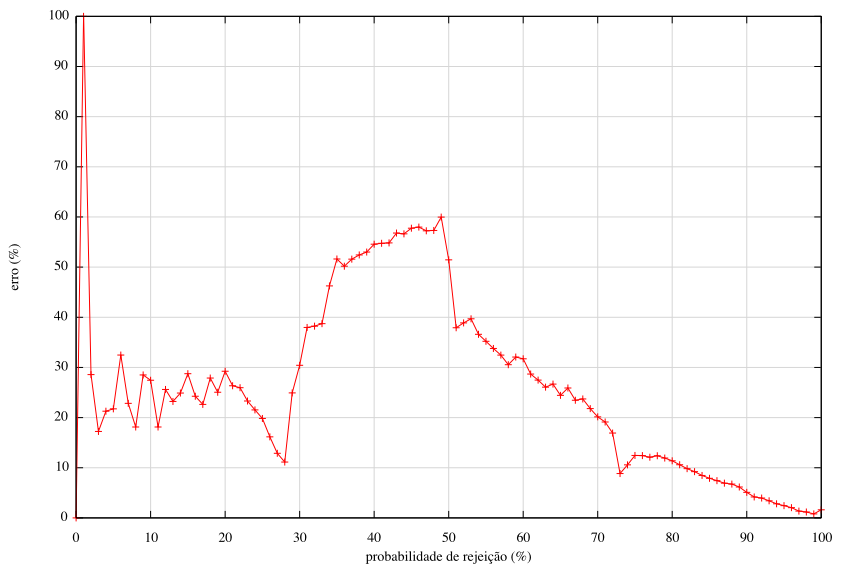
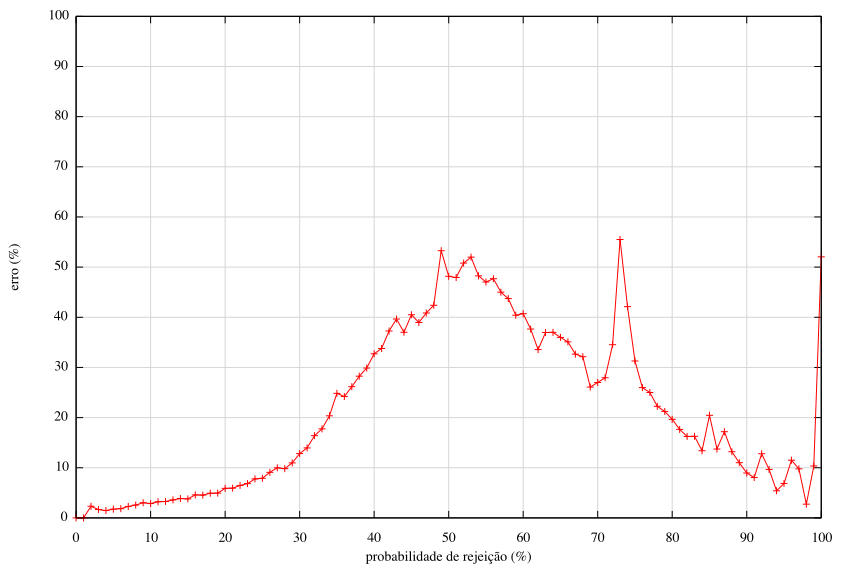
Em relação ao corpus 2, também tivemos uma melhora, porém bem mais sutil. O MCC foi de 0.514 para 0.522, um aumento de 1,5% apenas. As outras métricas também tiveram um pequeno aumento.

O grande problema da solução de utilizar classificadores de contexto é o consumo de recursos. Em relação a CPU, na pior hipótese, teríamos o consumo multiplicado por 2, pois para cada comentário seria necessário haver 2 classificadores (global e do grupo). Já em relação a memória, o consumo aumenta drasticamente pois é necessário armazenar além do classificador global um classificador para cada grupo. Assim, a implementação deste trabalho limitou o número de classificadores a 25.

### Terceira etapa

Até agora sempre utilizamos os classificadores de uma forma imperativa sobre sua resposta, ou seja, a resposta sobre a classificação de um comentário precisa ser aprovado ou rejeitado. Mas o objetivo deste trabalho não é realizar a classificação de forma automática, mas sim de apoiar o moderador. Portanto, se pudéssemos investigar os resultados das classificações mais a fundo e conseguíssemos desconsiderar as faixas que sabemos que nosso classificador tem um erro maior, poderíamos deixar estes comentários para o moderador humano e atuar somente nos comentários que a predição do classificador é mais confiável.

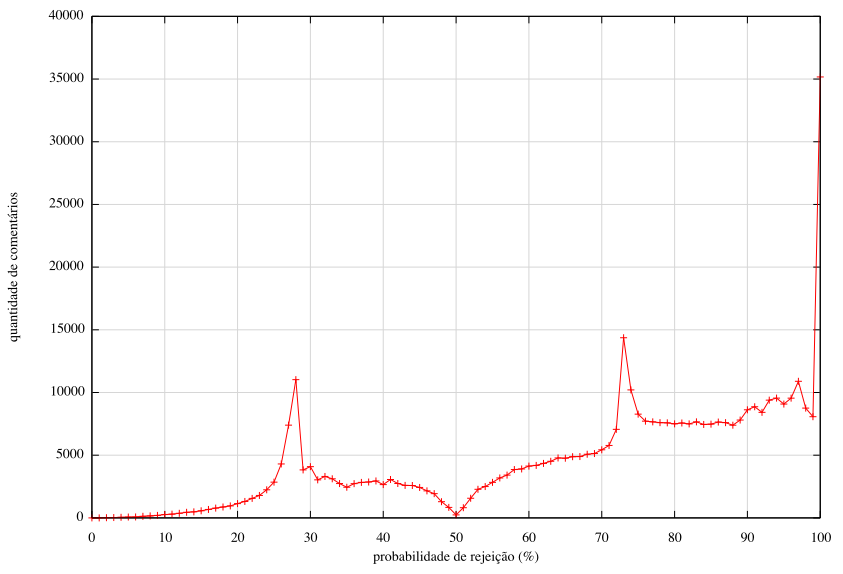
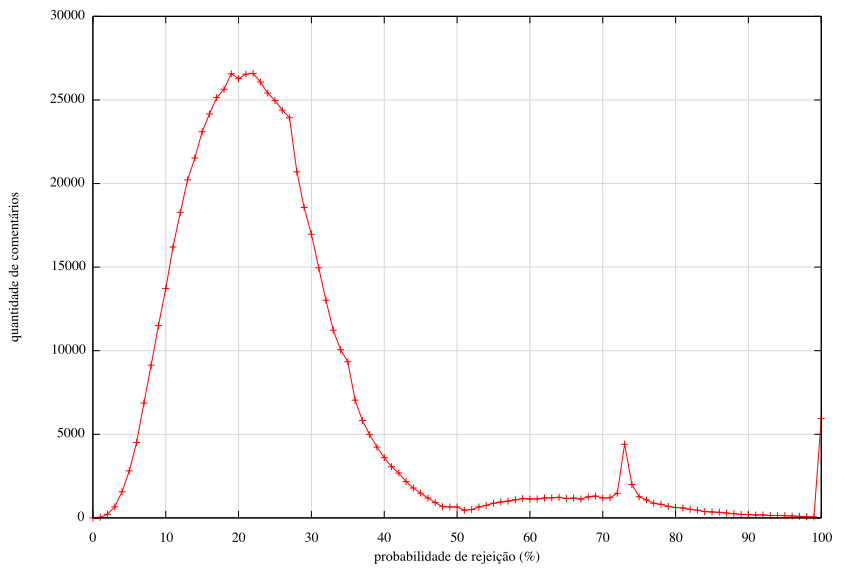
Ao analisar o funcionamento do SVM, podemos perceber que ele calcula, durante o treino, um hiperplano que separa os comentários aprovados dos comentários rejeitados (mais detalhes sobre o SVM na seção 3.2.3). Intuitivamente, podemos imaginar que os comentários que estão muito próximos ao hiperplano deveriam ter maior erro de classificação, pois um pequeno ruído adicionado por um atributo não relevante poderia alterar sua classificação. Para verificar se isto é verdade e determinar para quais valores isto ocorre, vamos analisar as probabilidades de cada comentário ser aprovado ou rejeitado, e não o resultado da classificação. Porém, como temos somente 2 classes, então as probabilidades são complementares, assim, no gráfico os comentários que tenham probabilidade menor que 50% para rejeição serão aprovados, enquanto que os comentários com probabilidade maior ou igual que 50% serão reprovados.



(I) (II)

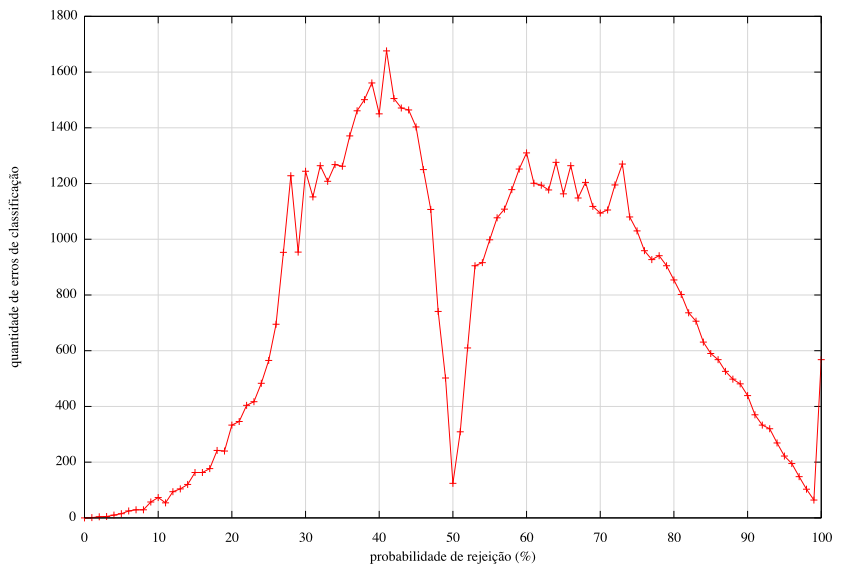
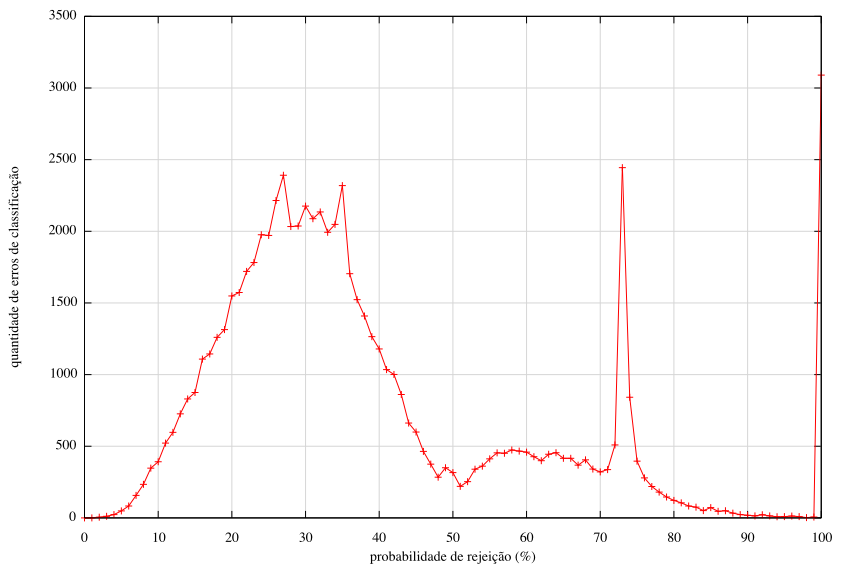
Figura 5 - Gráfico da probabilidade de rejeição por percentagem de erro

Observando os gráficos da figura 5 podemos verificar que a taxa de erro, relativo a probabilidade, é maior quando se está mais próximo da probabilidade de mudança de classe, que é de 50%. Do ponto do visto do SVM, são os comentários que estão mais próximos ao hiperplano que separa as duas classes. Porém, analisando a quantidade de comentários pela probabilidade de rejeição (figura 6), podemos perceber que poucos comentários foram classificados com probabilidade próximo a 50%. Como esta análise foi feita com base nos resultados do classificador de contexto e este classificador sempre privilegia os comentários que estão mais longe dos 50%, poucos comentários ainda ficaram nesta faixa, provavelmente os comentários dos grupos que continham poucos exemplos e por isto não possuíam um classificador especifico para o contexto, utilizando neste caso a resposta do classificador global.



(I) (II)

Figura 6 - Gráfico da quantidade de comentários por probabilidade de rejeição



(I) (II)

Figura 7 - Gráfico da contribuição do erro geral por probabilidade

Tanto o gráfico da figura 5 quanto da figura 6 tem um grande crescimento quando a probabilidade de rejeição é de 100%. Isto ocorre pois o moderador está configurado para rejeitar comentários com links, e-mails e códigos html com probabilidade de 100% (modo estrito), independente do resultado da classificação. Como os corpora possuem muitos comentários com estas características e nem todos foram reprovados, a taxa de erro na probabilidade 100% torna-se alta se comparada a taxa de erro do restante das probabilidades. Mesmo assim os moderadores configurados no modo estrito tem melhor performance que os moderadores que não utilizam esta configuração.

Observando o gráfico da figura 7 onde temos o erro total por probabilidade podemos perceber que, embora a taxa de erro por probabilidade seja alta próxima aos 50% de probabilidade de rejeição, a quantidade de comentários nesta faixa não é significativa. As outras faixas da figura 7, embora tenham uma quantidade absoluta de erro alta, não possuem um padrão que pudéssemos utilizar para evitar que a moderação automática ocorra. Assim, embora nossa hipótese da taxa de erro maior em torno dos 50%, a quantidade de comentários com esta probabilidade não é relevante para implementarmos algum mecanismo para ignorá-los e no restante das probabilidades não existe um padrão comum entre os 2 corpora.

## Discussão

Analisando as tabelas do anexo A podemos perceber que o SVM é o melhor algoritmo para a construção de modelos de predição para o problema de moderação de comentários, quando comparamos pelo MCC. É possível também perceber que o uso de n-grams de tamanho dois ou três melhoraram a performance do algoritmo. Porém, o resultado que chamou bastante atenção é o fato de que a escolha de métodos mais sofisticados para seleção de atributos como Correção de Palavras ou POS Tagging não gerou grandes variações quanto a parametrização do SVM para diferentes valores de C. Este resultado pode ser observado através dos desvios padrões calculados na tabela 4 e na tabela 5, ambas para o corpus 1. Um resultado análogo pode ser obtido utilizando o corpus 2.

Tabela 4 - Desvio padrão variando-se o parâmetro C (SVM)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Desvio padrão** | | | |
|  | MCC | F1 | Recall | Precisão |
| **SVM** | **0.018169108** | **0.038475612** | **5.70%** | **6.84%** |
| **Ngrams=1** | **0.012606096** | **0.037967964** | **5.32%** | **7.33%** |
| Corretor | 0.010139795 | 0.039590016 | 5.07% | 6.69% |
| POS (NB) | 0.010350067 | 0.047739702 | 8.10% | 11.58% |
| POS (rápido) | 0.012636272 | 0.045910110 | 5.72% | 7.20% |
| Simples | 0.011144125 | 0.041952234 | 5.80% | 7.95% |
| WNL | 0.011233123 | 0.040620605 | 5.39% | 6.99% |
| **Ngrams=2** | **0.006635894** | **0.028662409** | **5.38%** | **7.39%** |
| Corretor | 0.001690655 | 0.031657562 | 6.20% | 8.60% |
| POS (NB) | 0.001649556 | 0.030192703 | 5.72% | 8.03% |
| POS (rápido) | 0.002564504 | 0.031990605 | 6.17% | 8.46% |
| Simples | 0.002787638 | 0.032094573 | 6.12% | 8.37% |
| WNL | 0.001231306 | 0.032001168 | 5.90% | 7.92% |
| **Ngrams=3** | **0.008313110** | **0.007592435** | **0.71%** | **0.97%** |
| Simples | 0.000402527 | 0.000363514 | 0.03% | 0.04% |
| **Total** | **0.018169108** | **0.038475612** | **5.70%** | **6.84%** |

Tabela 5 - Desvio padrão variando-se o método de seleção de atributos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Desvio Padrão** | | | |
|  | MCC | F1 | Recall | Precisão |
| **SVM** | **0.018169108** | **0.038475612** | **5.70%** | **6.84%** |
| **Ngrams=1** | **0.012606096** | **0.037967964** | **5.32%** | **7.33%** |
| c=0.01 | 0.004564804 | 0.004201359 | 0.32% | 0.85% |
| c=1 | 0.01020117 | 0.006082274 | 1.62% | 3.46% |
| c=20 | 0.006887421 | 0.006703486 | 1.06% | 1.62% |
| c=50 | 0.013085057 | 0.005394757 | 1.98% | 4.03% |
| **Ngrams=2** | **0.006635894** | **0.028662409** | **5.38%** | **7.39%** |
| c=0.01 | 0.006037678 | 0.005142422 | 0.37% | 0.84% |
| c=1 | 0.007685122 | 0.006840849 | 0.66% | 0.86% |
| c=20 | 0.007079104 | 0.006663905 | 0.74% | 0.72% |
| c=50 | 0.007377702 | 0.006662267 | 0.78% | 0.94% |
| **Ngrams=3** | **0.008313110** | **0.007592435** | **0.71%** | **0.97%** |
| c=50 | 0.008825365 | 0.008155708 | 0.78% | 1.01% |
| **Total** | **0.018169108** | **0.038475612** | **5.70%** | **6.84%** |

Embora o uso de um corretor ortográfico não tenha reduzido o MCC, em relação ao recall houve uma melhora, porém, insignificante.

Uma explicação para o POS Tagging não ter melhorado o resultado, como ocorre em trabalhos de análise sentimental, é que a classe gramatical das palavras não é suficiente para auxiliar o algoritmo a distinguir entre um comentário aprovado ou rejeitado, sendo assim, adicioná-la apenas aumentou o número de atributos, gerando *overfitting*. Talvez combinado com um método para selecionar apenas os atributos (palavra mais classe gramatical) mais relevantes como feito em (VILELA, 2011) possa fazer valer a pena utilizar o POS Tagging.

Outro resultado importante pode ser observado com o uso do modo estrito, que bloqueia comentários com links, e-mails ou tags html. Embora a acurácia tenha se reduzido quando o utilizamos, juntamente com a precisão, as outras métricas aumentaram, especialmente o recall. A redução da acurácia com o uso do modo estrito acontece especialmente com mensagens de comentaristas especiais, que têm permissão de postar links ou uso de links internos. Para resolver isto seria importante também ter atributos sobre o usuário do comentários, assim usuários oficiais poderiam postar comentários mesmo que estes tenham links.

Embora o corpus 2 tenha mensagens reprovadas devido a falta de contexto das mesmas, este corpus apresentou melhor resultado para a construção de um classificador. Isto é medido através do valor absoluto do MCC. Conforme foi apresentado, quanto mais próximo o MCC for de 1 melhor é o preditor. Como neste corpus tivemos mais mensagens reprovadas que aprovadas, é possível que o maior trabalho do classificador seja de avaliar se a mensagem pertence ao contexto, tarefa que parece ser menos subjetiva que a moderação.

Por fim, os resultados mostraram a importância de utilizar métricas para avaliação dos resultados de acordo com o objetivo da classificação. Por exemplo, se estivéssemos interessados em utilizar o resultado deste trabalho para bloquear comentários indesejados de forma automática precisaríamos olhar para o resultado com melhor recall. Neste caso, teríamos para o corpus 2 o Naive Bayes com correção ortográfica como melhor resultado. O recall neste caso foi de 97,15%, ou seja, somente 2,85% dos comentários indesejados deste corpos foram aprovados incorretamente. Já para o corpus 1 o maior recall foi de 36,94%. Neste caso a probabilidade de um comentário indesejado ser aprovado é de 63,06%. Para este corpus a filtragem de comentários indesejados não seria muito eficiente.

# Conclusões e trabalhos futuros

Neste trabalho abordamos o problema da moderação de comentários, um problema análogo à classificação de mensagens de spam, porém a moderação automática de comentários tem um maior fator de subjetividade. Foram apresentados dois corpora construídos através de comentários feitos em sites da Globo.com e da TV Globo.

No Capítulo 2 mostramos como o problema de classificação foi solucionado em diversos trabalhos, tanto para filtros anti-spam quanto para sentiment analysis. Embora ambos trabalhos tenham interesse na classificação, o critério de avaliação dos algoritmos foi diferente, pois para anti-spam estava-se interessado em um método de classificação que minimizasse a quantidade de mensagens legítimas classificadas como spam enquanto nos trabalhos de sentiment analysis a acurácia era o critério mais importante.

No Capítulo 3 apresentamos os algoritmos Support Vector Machines, Naive Bayes e Boostexter. Apresentamos também métodos de extração de atributos que fazem a correção ortográfica do texto e a classificação gramatical. É apresentado ainda uma arquitetura que permite suportar a combinação dos diversos algoritmos e métodos.

No Capítulo 4 conduz um experimento sobre os dois corpora apresentados. Os critério de classificação já conhecidos são interpretados em face a este trabalho. Mostramos que para os corpora apresentados o Coeficiente de Correção é o melhor critério para se avaliar os classificadores, em vez da acurácia. Mostramos também que o recall mede a probabilidade de uma mensagem rejeitada ser aprovada e poderia ser utilizado caso o objetivo fosse de utilizar os resultados apresentados aqui para construção de um bloqueador automático de mensagens impróprias.

Ainda no Capítulo 4, nosso experimento mostrou que o SVM com saco de palavras e tri-grams apresentou o melhor resultado, com coeficiente de correlação igual a 0,375 para o corpus 1 e 0,514 para o corpus 2. Mostramos que o uso de classificadores especializados para grupos de comentário aumentaram o MCC para o primeiro corpus em quase 10% enquanto que para o segundo corpus o aumento foi de apenas 1,5%. Porém o uso de múltiplos classificadores aumenta consideravelmente o uso de recursos, tanto de memória quanto de cpu, por isto o número de especializações criadas precisa ser limitada. Tentamos ainda explorar a probabilidade de rejeição fornecida pelo SVM, em vez de simplesmente a classe de maior probabilidade, definindo assim uma faixa onde o erro fosse muito alto e o uso de classificador automático não fosse útil, porém não tivemos sucesso pois a quantidade de comentários classificados nestas faixas é irrelevante.

Para conseguir atingir melhores resultados abaixo listo algumas ideias que podem ser exploradas:

* Explorar melhor a ideia de classificadores de contexto, melhorando o algoritmo de escolha entre eles e reduzir o consumo de recursos.
* Utilizar algum método de seleção dos atributos, como os apresentados por (VILELA, 2011).
* Construir um corpus com mais informações sobre as mensagens, como dados do usuário, IP do usuário, quem foi o moderador, entre outros.

# Bibliografia

ALUÍSIO, S. M. et al. An Account of the Challenge of Tagging a Reference Corpus for Brazilian Portuguese. **Computational Processing of the Portuguese Language**, v. 2721, p. 194, 2003.

ALVIM, L. G. M. et al. **Sentiment of Financial News:** A Natural Language Processing Approach. Proceedings of the 1st Workshop on Natural Language Processing Tools Applied to Discourse Analysis in Psychology. Buenos Aires: [s.n.]. 2010. 10-14 Mai 2010.

ANDROUTSOPOULOS, I. et al. **An evaluation of Naive Bayesian anti-spam filtering**. Proceedings of the workshop on Machine Learning in the New Information Age and 11th European Conference on Machine Learning. Barcelona, Spain: ECML 2000. 2000. p. 9-17.

ANDROUTSOPOULOS, I. et al. **An experimental comparison of naive Bayesian and keyword-based anti-spam filtering with personal e-mail messages**. Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information. New York, NY, USA: ACM. 2000. p. 160-167.

ATKINSON, K. GNU Aspell. **GNU Aspell**, 2004. Disponivel em: <http://aspell.net/>. Acesso em: 20 maio 2011.

BERGER, A. L.; PIETRA, S. A. D.; PIETRA, V. J. D. A maximum entropy approach to natural language processing. **Computational Linguistics**, Cambridge, MA, USA, v. 22, n. 1, p. 39-71, Mar 1996. ISSN 0891-2017.

BIRD, S.; LOPER, E. Natural Language Toolkit. **Natural Language Toolkit**, 2011. Disponivel em: <http://www.nltk.org/>. Acesso em: 28 Jul 2011.

CARRERAS, X.; MÀRQUEZ, L. **Boosting Trees for Anti-Spam Email Filtering**. Proceedings of RANLP-2001. Bulgaria: [s.n.]. 2001. p. 58-64.

CHURCH, K. W.; HANKS, P. Word association norms, mutual information, and lexicography. **Computational Linguistics**, Cambridge, v. 16, n. 1, p. 22-29, Mar 1990. ISSN 0891-2017.

DOMAINTOOLS.COM. Daily DNS Changes and Web Hosting Activity by DailyChanges.com. **DailyChanges.com**, 2011. Disponivel em: <http://www.dailychanges.com/>. Acesso em: 19 Jan 2011.

DRUCKER, H.; WU, D.; VAPNIK, V. N. Support Vector Machines for Spam Categorization. **Neural Networks, IEEE Transactions on**, v. 10, n. 5, p. 1048 -1054, Set 1999. ISSN 1045-9227.

FAN, R.-E. et al. LIBLINEAR: A Library for Large Linear Classification. **The Journal of Machine Learning Research**, v. 9, p. 1871-1874, Jun 2008. ISSN 1532-4435.

FAVRE, B.; HAKKANI, D. Icsiboost. **Icsiboost - Open-source implementation of Boostexter (Adaboost based classifier)**, 2011. Disponivel em: <http://code.google.come/p/icsiboost>. Acesso em: 28 jul. 2011.

GAMMA, E. et al. **Design Patterns:** Elements of Reusable Object-Oriented Software. 1st Edition. ed. [S.l.]: Addison-Wesley Professional, 1994. ISBN 0201633612.

GNU ASPELL. Spell Checker Test Kernel Results. **GNU Aspell**, 2011. Disponivel em: <http://aspell.net/test/cur/>. Acesso em: 16 Jul 2011.

JOACHIMS, T. Text categorization with Support Vector Machines: Learning with many relevant features. In: JOACHIMS, T. **10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings**. Dortmund: Springer Berlin / Heidelberg, v. 1398, 1998. p. 137-142.

LAI, C.-C. An empirical study of three machine learning methods for spam filtering. **Know.-Based Syst.**, Amsterdam, v. 20, n. 3, p. 249-254, Abr 2007. ISSN 0950-7051.

LEVENSHTEIN, V. I. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. **Soviet Physics Doklady**, v. 10, p. 707-710, feb 1966.

LEWIS, D. Naive (Bayes) at forty: The independence assumption in information retrieval. In: NÉDELLEC, C. A. R. C. **10th European Conference on Machine Learning Chemnitz, Germany, April 21–23, 1998 Proceedings**. New Jersey: Springer Berlin / Heidelberg, v. 1398, 1998. p. 4-15. ISBN 978-3-540-64417-0.

MINIWATTS MARKETING GROUP. World Internet Usage Statistics News and World Population Stats. **Internet World Stats**, 2011. Disponivel em: <http://www.internetworldstats.com/stats.htm>. Acesso em: 19 Jan 2011.

PANG, B.; LEE, L.; VAITHYANATHAN, S. Thumbs up? Sentiment Classification using Machine Learning Techniques. **Proceeding of the conference on empirical methods in natural language processing (EMNLP, 2002)**, Philadelphia, PA, USA, v. 10, p. 79-86, 6-7 Jul 2002.

SAKKIS, G. et al. **Stacking classifiers for anti-spam filtering of e-mail**. Proceedings of "Empirical Methods in Natural Language Processing". Pittsburgh, PA: EMNLP 2001. 2001. p. 44-50.

SCHAPIRE, R. E.; SINGER, Y. BoosTexter: A boosting-based system for text categorization. **Machine Learning**, v. 2/3, n. 39, p. 135-168, 2000.

TURNEY, P. D. **Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews**. Proceedings of the 40th Annual Meeting on Association for Computational Linguistics. Philadelphia, Pennsylvania: Association for Computational Linguistics. 2002. p. 417-424.

VAPNIK, V.; LERNER, A. Pattern recognition using generalized portrait method. **Automation and Remote Control**, v. 24, p. 774-780, 1963.

VILELA, P. D. C. S. **Classificação de Sentimento para Notícias sobre a Petrobras no Mercado Financeiro**. Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro, p. 52. 2011.

WIKIMEDIA FOUNDATION. Cross-site scripting. **Wikipedia:** The Free Encyclopedia, 2011. Disponivel em: <http://en.wikipedia.org/wiki/Cross-site\_scripting>. Acesso em: 20 Maio 2011.

YONG-FENG, S.; YAN-PING, Z. Comparison of text categorization algorithms. **Wuhan University Journal of Natural Sciences**, v. 9, n. 5, p. 798-804, 2004.

# 

# Apêndice A – Resultados da etapa I dos experimentos

As tabelas abaixo listam os resultados obtidos experimentalmente para a etapa I, com os corpora I e II. Os experimentos conduzem combinações de resultados que foram julgadas relevantes, não possuem todas as combinações possíveis.

Algumas observações sobre os resultados são necessárias:

* Ordenação decrescente pelo MCC.
* Extratores:
  + Simples: somente palavras e números.
  + WNL: palavras, números e sinais de pontuação.
  + Corretor: palavras e números, mas com correção das palavras.
  + POS (rápido): com classificação gramatical rápida (ver capítulo 4).
  + POS (NB): com classificação gramatical utilizando o Naive Bayes (ver capítulo 4).
* Colunas:
  + Algoritmo: nome do algoritmo utilizado
  + Param: para o SVM e o Boostexter alguns parâmetros precisam ser configurados. No caso do SVM o parâmetro de margin C e para o Boostexter o número de iterações.
  + N-grams: número de n-grams
  + FE: método para extrair os atributos dos comentários
  + Strict: Indica se os comentários com e-mails, links ou código javascript/html foram rejeitados independente do classificador
  + RC: Quantidade de comentários reprovados corretamente
  + AC: Quantidade de comentários aprovados corretamente
  + RI: Quantidade de comentários reprovados incorretamente
  + AI: Quantidade de comentários aprovados incorretamente
  + Acurácia: métrica de acurácia
  + Precisão: métrica de precisão
  + Recall: métrica de recall
  + F1: F-measure ajustado para o parâmetro =1 (peso igual para as 2 classes)
  + MCC: Coeficiente de correlação
* Resultados para o Corpus 1

| Algoritmo | Param | Ngrams | FE | Strict | RC | AC | RI | AI | Acurácia | Precisão | Recall | F1 | MCC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SVM | c=20 | 3 | Simples | Sim | 28,540 | 551,043 | 22,778 | 55,044 | 88.16% | 55.61% | 34.15% | 0.423 | 0.375 |
| SVM | c=50 | 3 | Simples | Sim | 28,503 | 551,017 | 22,804 | 55,081 | 88.15% | 55.55% | 34.10% | 0.423 | 0.374 |
| SVM | c=50 | 3 | WNL | Sim | 28,147 | 551,505 | 22,316 | 55,437 | 88.17% | 55.78% | 33.68% | 0.420 | 0.373 |
| SVM | c=50 | 3 | Correção | Sim | 28,781 | 550,068 | 23,753 | 54,803 | 88.05% | 54.79% | 34.43% | 0.423 | 0.372 |
| SVM | c=50 | 3 | Simples | Não | 27,361 | 552,976 | 20,845 | 56,223 | 88.28% | 56.76% | 32.73% | 0.415 | 0.372 |
| SVM | c=50 | 3 | POS (rápido) | Sim | 28,647 | 549,909 | 23,912 | 54,937 | 88.01% | 54.50% | 34.27% | 0.421 | 0.370 |
| SVM | c=0.01 | 2 | Simples | Não | 19,258 | 565,498 | 8,323 | 64,326 | 88.95% | 69.82% | 23.04% | 0.346 | 0.359 |
| SVM | c=1 | 2 | Simples | Sim | 30,124 | 543,986 | 29,835 | 53,460 | 87.33% | 50.24% | 36.04% | 0.420 | 0.357 |
| SVM | c=50 | 2 | WNL | Sim | 30,163 | 543,742 | 30,079 | 53,421 | 87.30% | 50.07% | 36.09% | 0.419 | 0.356 |
| SVM | c=1 | 2 | WNL | Sim | 29,979 | 544,151 | 29,670 | 53,605 | 87.33% | 50.26% | 35.87% | 0.419 | 0.356 |
| SVM | c=20 | 2 | WNL | Sim | 30,393 | 542,781 | 31,040 | 53,191 | 87.19% | 49.47% | 36.36% | 0.419 | 0.354 |
| SVM | c=0.01 | 2 | WNL | Sim | 20,326 | 563,241 | 10,580 | 63,258 | 88.77% | 65.77% | 24.32% | 0.355 | 0.354 |
| SVM | c=0.01 | 2 | Simples | Sim | 20,288 | 563,293 | 10,528 | 63,296 | 88.77% | 65.84% | 24.27% | 0.355 | 0.354 |
| SVM | c=1 | 2 | POS (rápido) | Sim | 30,257 | 542,886 | 30,935 | 53,327 | 87.18% | 49.45% | 36.20% | 0.418 | 0.353 |
| SVM | c=0.01 | 2 | Correção | Sim | 20,269 | 563,247 | 10,574 | 63,315 | 88.76% | 65.72% | 24.25% | 0.354 | 0.353 |
| SVM | c=1 | 2 | Simples | Não | 29,172 | 545,418 | 28,403 | 54,412 | 87.40% | 50.67% | 34.90% | 0.413 | 0.353 |
| SVM | c=50 | 3 | POS (NB) | Sim | 27,159 | 549,960 | 23,861 | 56,425 | 87.79% | 53.23% | 32.49% | 0.404 | 0.353 |
| SVM | c=20 | 2 | Simples | Sim | 30,665 | 541,713 | 32,108 | 52,919 | 87.07% | 48.85% | 36.69% | 0.419 | 0.352 |
| SVM | c=1 | 2 | Correção | Sim | 30,309 | 542,444 | 31,377 | 53,275 | 87.12% | 49.13% | 36.26% | 0.417 | 0.352 |
| SVM | c=0.01 | 2 | POS (rápido) | Sim | 20,181 | 563,170 | 10,651 | 63,403 | 88.74% | 65.45% | 24.14% | 0.353 | 0.351 |
| SVM | c=50 | 2 | Correção | Sim | 30,878 | 540,689 | 33,132 | 52,706 | 86.94% | 48.24% | 36.94% | 0.418 | 0.350 |
| SVM | c=50 | 2 | Simples | Sim | 30,721 | 541,065 | 32,756 | 52,863 | 86.98% | 48.40% | 36.75% | 0.418 | 0.350 |
| SVM | c=20 | 2 | Simples | Não | 29,402 | 544,176 | 29,645 | 54,182 | 87.25% | 49.79% | 35.18% | 0.412 | 0.350 |
| SVM | c=50 | 2 | Simples | Não | 29,442 | 543,979 | 29,842 | 54,142 | 87.22% | 49.66% | 35.22% | 0.412 | 0.349 |
| SVM | c=20 | 2 | Correção | Sim | 30,686 | 540,924 | 32,897 | 52,898 | 86.95% | 48.26% | 36.71% | 0.417 | 0.349 |
| Naive Bayes |  | 1 | Simples | Sim | 26,380 | 550,920 | 22,901 | 57,204 | 87.81% | 53.53% | 31.56% | 0.397 | 0.349 |
| SVM | c=50 | 2 | POS (rápido) | Sim | 30,653 | 540,887 | 32,934 | 52,931 | 86.94% | 48.21% | 36.67% | 0.417 | 0.349 |
| SVM | c=20 | 2 | POS (rápido) | Sim | 30,583 | 540,814 | 33,007 | 53,001 | 86.92% | 48.09% | 36.59% | 0.416 | 0.348 |
| SVM | c=1 | 1 | WNL | Sim | 24,746 | 554,103 | 19,718 | 58,838 | 88.05% | 55.65% | 29.61% | 0.387 | 0.347 |
| Naive Bayes |  | 1 | WNL | Sim | 25,706 | 551,926 | 21,895 | 57,878 | 87.87% | 54.00% | 30.75% | 0.392 | 0.346 |
| SVM | c=1 | 1 | Simples | Sim | 24,953 | 553,156 | 20,665 | 58,631 | 87.94% | 54.70% | 29.85% | 0.386 | 0.344 |
| SVM | c=1 | 1 | Simples | Não | 24,048 | 554,585 | 19,236 | 59,536 | 88.02% | 55.56% | 28.77% | 0.379 | 0.341 |
| SVM | c=0.01 | 2 | POS (NB) | Sim | 19,590 | 562,723 | 11,098 | 63,994 | 88.58% | 63.84% | 23.44% | 0.343 | 0.340 |
| SVM | c=20 | 1 | WNL | Sim | 25,583 | 550,912 | 22,909 | 58,001 | 87.69% | 52.76% | 30.61% | 0.387 | 0.339 |
| SVM | c=1 | 2 | POS (NB) | Sim | 28,967 | 542,698 | 31,123 | 54,617 | 86.96% | 48.21% | 34.66% | 0.403 | 0.338 |
| SVM | c=20 | 2 | POS (NB) | Sim | 29,221 | 541,674 | 32,147 | 54,363 | 86.84% | 47.62% | 34.96% | 0.403 | 0.336 |
| SVM | c=1 | 1 | POS (rápido) | Sim | 23,819 | 554,175 | 19,646 | 59,765 | 87.92% | 54.80% | 28.50% | 0.375 | 0.336 |
| SVM | c=50 | 2 | POS (NB) | Sim | 29,271 | 541,533 | 32,288 | 54,313 | 86.83% | 47.55% | 35.02% | 0.403 | 0.336 |
| SVM | c=1 | 1 | Correção | Sim | 23,803 | 554,159 | 19,662 | 59,781 | 87.92% | 54.76% | 28.48% | 0.375 | 0.336 |
| SVM | c=50 | 1 | WNL | Sim | 26,300 | 548,142 | 25,679 | 57,284 | 87.38% | 50.60% | 31.47% | 0.388 | 0.333 |
| SVM | c=50 | 1 | POS (rápido) | Sim | 24,631 | 551,081 | 22,740 | 58,953 | 87.57% | 52.00% | 29.47% | 0.376 | 0.329 |
| SVM | c=20 | 1 | Simples | Sim | 26,410 | 546,932 | 26,889 | 57,174 | 87.21% | 49.55% | 31.60% | 0.386 | 0.328 |
| SVM | c=50 | 1 | Simples | Sim | 26,440 | 546,546 | 27,275 | 57,144 | 87.16% | 49.22% | 31.63% | 0.385 | 0.327 |
| SVM | c=50 | 1 | Correção | Sim | 24,839 | 550,276 | 23,545 | 58,745 | 87.48% | 51.34% | 29.72% | 0.376 | 0.327 |
| SVM | c=20 | 1 | Correção | Sim | 24,647 | 550,637 | 23,184 | 58,937 | 87.51% | 51.53% | 29.49% | 0.375 | 0.326 |
| SVM | c=20 | 1 | Simples | Não | 25,539 | 548,063 | 25,758 | 58,045 | 87.25% | 49.79% | 30.55% | 0.379 | 0.324 |
| SVM | c=50 | 1 | Simples | Não | 25,736 | 547,471 | 26,350 | 57,848 | 87.19% | 49.41% | 30.79% | 0.379 | 0.323 |
| SVM | c=1 | 1 | POS (NB) | Sim | 27,114 | 543,612 | 30,209 | 56,470 | 86.81% | 47.30% | 32.44% | 0.385 | 0.321 |
| SVM | c=0.01 | 1 | WNL | Sim | 16,626 | 565,391 | 8,430 | 66,958 | 88.53% | 66.36% | 19.89% | 0.306 | 0.321 |
| SVM | c=0.01 | 1 | Simples | Não | 15,103 | 567,627 | 6,194 | 68,481 | 88.64% | 70.92% | 18.07% | 0.288 | 0.320 |
| SVM | c=0.01 | 1 | Simples | Sim | 16,338 | 565,495 | 8,326 | 67,246 | 88.50% | 66.24% | 19.55% | 0.302 | 0.317 |
| SVM | c=0.01 | 1 | POS (rápido) | Sim | 15,976 | 565,455 | 8,366 | 67,608 | 88.44% | 65.63% | 19.11% | 0.296 | 0.311 |
| SVM | c=0.01 | 1 | Correção | Sim | 15,996 | 565,411 | 8,410 | 67,588 | 88.44% | 65.54% | 19.14% | 0.296 | 0.311 |
| SVM | c=0.01 | 1 | POS (NB) | Sim | 16,300 | 564,742 | 9,079 | 67,284 | 88.38% | 64.23% | 19.50% | 0.299 | 0.310 |
| SVM | c=50 | 1 | POS (NB) | Sim | 28,754 | 534,235 | 39,586 | 54,830 | 85.64% | 42.07% | 34.40% | 0.379 | 0.300 |
| Boostexter | i=25 | 1 | Simples | Sim | 12,337 | 566,268 | 7,553 | 71,247 | 88.01% | 62.03% | 14.76% | 0.238 | 0.261 |
| Boostexter | i=25 | 2 | Simples | Sim | 11,794 | 566,584 | 7,237 | 71,790 | 87.98% | 61.97% | 14.11% | 0.230 | 0.255 |
| Boostexter | i=25 | 1 | Correção | Sim | 11,626 | 566,505 | 7,316 | 71,958 | 87.94% | 61.38% | 13.91% | 0.227 | 0.252 |
| Boostexter | i=25 | 2 | Correção | Sim | 11,559 | 566,526 | 7,295 | 72,025 | 87.93% | 61.31% | 13.83% | 0.226 | 0.251 |
| Boostexter | i=25 | 1 | POS (rápido) | Sim | 11,405 | 566,627 | 7,194 | 72,179 | 87.93% | 61.32% | 13.64% | 0.223 | 0.249 |
| Boostexter | i=25 | 2 | POS (rápido) | Sim | 11,353 | 566,626 | 7,195 | 72,231 | 87.92% | 61.21% | 13.58% | 0.222 | 0.248 |
| Boostexter | i=25 | 2 | POS (NB) | Sim | 10,265 | 567,137 | 6,684 | 73,319 | 87.83% | 60.56% | 12.28% | 0.204 | 0.234 |
| Boostexter | i=25 | 1 | POS (NB) | Sim | 10,184 | 567,193 | 6,628 | 73,400 | 87.83% | 60.58% | 12.18% | 0.203 | 0.233 |

* Resultados para o Corpus 2

| Algoritmo | Param | Ngrams | FE | Strict | RC | AC | RI | AI | Acurácia | Precisão | Recall | F1 | MCC |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| SVM | c=50 | 3 | WNL | Sim | 312,937 | 63,585 | 39,269 | 35,418 | 83.45% | 88.85% | 89.83% | 0.893 | 0.524 |
| SVM | c=0.1 | 2 | WNL | Sim | 317,159 | 60,427 | 42,427 | 31,196 | 83.68% | 88.20% | 91.04% | 0.896 | 0.519 |
| SVM | c=1 | 2 | WNL | Sim | 312,067 | 63,419 | 39,435 | 36,288 | 83.22% | 88.78% | 89.58% | 0.892 | 0.518 |
| SVM | c=20 | 3 | Simples | Sim | 313,858 | 61,857 | 40,997 | 34,497 | 83.27% | 88.45% | 90.10% | 0.893 | 0.514 |
| SVM | c=10 | 2 | WNL | Sim | 310,535 | 63,796 | 39,058 | 37,820 | 82.96% | 88.83% | 89.14% | 0.890 | 0.514 |
| SVM | c=50 | 3 | Simples | Sim | 314,048 | 61,673 | 41,181 | 34,307 | 83.27% | 88.41% | 90.15% | 0.893 | 0.514 |
| SVM | c=50 | 2 | WNL | Sim | 310,488 | 63,735 | 39,119 | 37,867 | 82.94% | 88.81% | 89.13% | 0.890 | 0.513 |
| SVM | c=5 | 2 | WNL | Sim | 310,241 | 63,874 | 38,980 | 38,114 | 82.91% | 88.84% | 89.06% | 0.889 | 0.513 |
| SVM | c=20 | 2 | WNL | Sim | 309,871 | 63,954 | 38,900 | 38,484 | 82.85% | 88.85% | 88.95% | 0.889 | 0.512 |
| SVM | c=50 | 3 | Simples | Sim | 313,468 | 61,812 | 41,042 | 34,887 | 83.17% | 88.42% | 89.99% | 0.892 | 0.512 |
| SVM | c=50 | 3 | POS (NB) | Sim | 315,511 | 60,497 | 42,357 | 32,844 | 83.33% | 88.16% | 90.57% | 0.894 | 0.512 |
| SVM | c=0.1 | 2 | Simples | Não | 318,972 | 58,158 | 44,696 | 29,383 | 83.58% | 87.71% | 91.57% | 0.896 | 0.510 |
| SVM | c=0.1 | 2 | POS (NB) | Sim | 319,061 | 58,061 | 44,793 | 29,294 | 83.58% | 87.69% | 91.59% | 0.896 | 0.510 |
| SVM | c=1 | 2 | Simples | Não | 311,620 | 62,567 | 40,287 | 36,735 | 82.93% | 88.55% | 89.45% | 0.890 | 0.509 |
| SVM | c=1 | 2 | Simples | Sim | 311,901 | 62,307 | 40,547 | 36,454 | 82.93% | 88.50% | 89.54% | 0.890 | 0.508 |
| SVM | c=0.1 | 2 | Corretor | Sim | 318,375 | 58,273 | 44,581 | 29,980 | 83.48% | 87.72% | 91.39% | 0.895 | 0.508 |
| SVM | c=0.1 | 2 | POS (rápido) | Sim | 318,674 | 58,058 | 44,796 | 29,681 | 83.49% | 87.68% | 91.48% | 0.895 | 0.508 |
| SVM | c=50 | 3 | POS (rápido) | Sim | 311,910 | 62,110 | 40,744 | 36,445 | 82.89% | 88.45% | 89.54% | 0.890 | 0.507 |
| SVM | c=1 | 2 | POS (rápido) | Sim | 312,153 | 61,933 | 40,921 | 36,202 | 82.91% | 88.41% | 89.61% | 0.890 | 0.507 |
| SVM | c=50 | 3 | Corretor | Sim | 311,970 | 61,965 | 40,889 | 36,385 | 82.87% | 88.41% | 89.56% | 0.890 | 0.506 |
| SVM | c=5 | 2 | Simples | Não | 309,673 | 63,127 | 39,727 | 38,682 | 82.62% | 88.63% | 88.90% | 0.888 | 0.505 |
| SVM | c=5 | 2 | Simples | Sim | 310,304 | 62,736 | 40,118 | 38,051 | 82.68% | 88.55% | 89.08% | 0.888 | 0.504 |
| SVM | c=10 | 2 | Simples | Não | 309,344 | 63,261 | 39,593 | 39,011 | 82.58% | 88.65% | 88.80% | 0.887 | 0.504 |
| SVM | c=1 | 2 | Corretor | Sim | 311,160 | 62,189 | 40,665 | 37,195 | 82.74% | 88.44% | 89.32% | 0.889 | 0.504 |
| SVM | c=0.1 | 2 | Simples | Sim | 319,351 | 57,115 | 45,739 | 29,004 | 83.43% | 87.47% | 91.67% | 0.895 | 0.504 |
| SVM | c=50 | 2 | Simples | Não | 308,868 | 63,432 | 39,422 | 39,487 | 82.51% | 88.68% | 88.66% | 0.887 | 0.503 |
| SVM | c=50 | 2 | Simples | Não | 308,663 | 63,545 | 39,309 | 39,692 | 82.49% | 88.70% | 88.61% | 0.887 | 0.503 |
| SVM | c=20 | 2 | Simples | Não | 308,975 | 63,328 | 39,526 | 39,380 | 82.51% | 88.66% | 88.70% | 0.887 | 0.503 |
| SVM | c=50 | 2 | Simples | Sim | 309,160 | 63,158 | 39,696 | 39,195 | 82.52% | 88.62% | 88.75% | 0.887 | 0.502 |
| SVM | c=10 | 2 | Simples | Sim | 309,258 | 63,073 | 39,781 | 39,097 | 82.52% | 88.60% | 88.78% | 0.887 | 0.502 |
| SVM | c=1 | 2 | POS (NB) | Sim | 311,545 | 61,738 | 41,116 | 36,810 | 82.73% | 88.34% | 89.43% | 0.889 | 0.502 |
| SVM | c=50 | 2 | Simples | Sim | 309,193 | 62,967 | 39,887 | 39,162 | 82.48% | 88.57% | 88.76% | 0.887 | 0.501 |
| SVM | c=5 | 2 | Corretor | Sim | 309,596 | 62,676 | 40,178 | 38,759 | 82.51% | 88.51% | 88.87% | 0.887 | 0.501 |
| SVM | c=5 | 2 | POS (rápido) | Sim | 309,639 | 62,623 | 40,231 | 38,716 | 82.50% | 88.50% | 88.89% | 0.887 | 0.500 |
| SVM | c=20 | 2 | Simples | Sim | 308,810 | 62,986 | 39,868 | 39,545 | 82.40% | 88.57% | 88.65% | 0.886 | 0.499 |
| SVM | c=10 | 2 | POS (rápido) | Sim | 309,055 | 62,786 | 40,068 | 39,300 | 82.41% | 88.52% | 88.72% | 0.886 | 0.499 |
| SVM | c=50 | 2 | POS (rápido) | Sim | 308,999 | 62,790 | 40,064 | 39,356 | 82.40% | 88.52% | 88.70% | 0.886 | 0.499 |
| SVM | c=20 | 2 | POS (rápido) | Sim | 308,763 | 62,894 | 39,960 | 39,592 | 82.37% | 88.54% | 88.63% | 0.886 | 0.498 |
| SVM | c=20 | 2 | Corretor | Sim | 308,547 | 62,975 | 39,879 | 39,808 | 82.34% | 88.55% | 88.57% | 0.886 | 0.498 |
| SVM | c=10 | 2 | Corretor | Sim | 308,340 | 63,030 | 39,824 | 40,015 | 82.31% | 88.56% | 88.51% | 0.885 | 0.498 |
| SVM | c=50 | 2 | Corretor | Sim | 307,624 | 63,289 | 39,565 | 40,731 | 82.20% | 88.60% | 88.31% | 0.885 | 0.496 |
| SVM | c=5 | 2 | POS (NB) | Sim | 308,337 | 62,499 | 40,355 | 40,018 | 82.19% | 88.43% | 88.51% | 0.885 | 0.493 |
| SVM | c=10 | 2 | POS (NB) | Sim | 308,438 | 62,425 | 40,429 | 39,917 | 82.19% | 88.41% | 88.54% | 0.885 | 0.493 |
| SVM | c=50 | 2 | POS (NB) | Sim | 307,922 | 62,375 | 40,479 | 40,433 | 82.07% | 88.38% | 88.39% | 0.884 | 0.490 |
| SVM | c=20 | 2 | POS (NB) | Sim | 307,438 | 62,520 | 40,334 | 40,917 | 81.99% | 88.40% | 88.25% | 0.883 | 0.489 |
| SVM | c=5 | 1 | WNL | Sim | 313,086 | 57,623 | 45,231 | 35,269 | 82.16% | 87.38% | 89.88% | 0.886 | 0.476 |
| SVM | c=50 | 1 | WNL | Sim | 313,375 | 57,335 | 45,519 | 34,980 | 82.16% | 87.32% | 89.96% | 0.886 | 0.475 |
| SVM | c=20 | 1 | WNL | Sim | 313,817 | 56,863 | 45,991 | 34,538 | 82.15% | 87.22% | 90.09% | 0.886 | 0.474 |
| SVM | c=1 | 1 | WNL | Sim | 314,769 | 55,848 | 47,006 | 33,586 | 82.14% | 87.01% | 90.36% | 0.887 | 0.470 |
| SVM | c=10 | 1 | WNL | Sim | 314,101 | 56,211 | 46,643 | 34,254 | 82.07% | 87.07% | 90.17% | 0.886 | 0.470 |
| SVM | c=1 | 1 | POS (NB) | Sim | 312,082 | 57,384 | 45,470 | 36,273 | 81.88% | 87.28% | 89.59% | 0.884 | 0.469 |
| SVM | c=5 | 1 | Simples | Sim | 310,683 | 58,118 | 44,736 | 37,672 | 81.74% | 87.41% | 89.19% | 0.883 | 0.469 |
| SVM | c=50 | 1 | Simples | Sim | 310,292 | 58,342 | 44,512 | 38,063 | 81.70% | 87.45% | 89.07% | 0.883 | 0.469 |
| SVM | c=0.01 | 2 | WNL | Sim | 322,914 | 50,492 | 52,362 | 25,441 | 82.76% | 86.05% | 92.70% | 0.892 | 0.469 |
| SVM | c=20 | 1 | Simples | Sim | 310,776 | 57,970 | 44,884 | 37,579 | 81.72% | 87.38% | 89.21% | 0.883 | 0.468 |
| SVM | c=5 | 1 | POS (NB) | Sim | 309,001 | 58,965 | 43,889 | 39,354 | 81.55% | 87.56% | 88.70% | 0.881 | 0.468 |
| SVM | c=50 | 1 | Simples | Não | 309,840 | 58,461 | 44,393 | 38,515 | 81.63% | 87.47% | 88.94% | 0.882 | 0.468 |
| SVM | c=10 | 1 | POS (NB) | Sim | 308,286 | 59,291 | 43,563 | 40,069 | 81.46% | 87.62% | 88.50% | 0.881 | 0.467 |
| SVM | c=20 | 1 | POS (NB) | Sim | 307,931 | 59,461 | 43,393 | 40,424 | 81.42% | 87.65% | 88.40% | 0.880 | 0.467 |
| SVM | c=50 | 1 | POS (NB) | Sim | 307,727 | 59,534 | 43,320 | 40,628 | 81.39% | 87.66% | 88.34% | 0.880 | 0.467 |
| SVM | c=10 | 1 | Simples | Não | 311,204 | 57,458 | 45,396 | 37,151 | 81.71% | 87.27% | 89.34% | 0.883 | 0.466 |
| SVM | c=10 | 1 | Simples | Sim | 310,684 | 57,650 | 45,204 | 37,671 | 81.63% | 87.30% | 89.19% | 0.882 | 0.465 |
| SVM | c=20 | 1 | Simples | Não | 310,681 | 57,540 | 45,314 | 37,674 | 81.61% | 87.27% | 89.19% | 0.882 | 0.464 |
| SVM | c=5 | 1 | Simples | Não | 311,251 | 57,134 | 45,720 | 37,104 | 81.64% | 87.19% | 89.35% | 0.883 | 0.463 |
| Naive Bayes |  | 2 | POS (NB) | Sim | 310,835 | 56,759 | 46,095 | 37,520 | 81.47% | 87.09% | 89.23% | 0.881 | 0.458 |
| SVM | c=1 | 1 | Simples | Não | 313,711 | 55,061 | 47,793 | 34,644 | 81.73% | 86.78% | 90.05% | 0.884 | 0.458 |
| SVM | c=1 | 1 | Simples | Sim | 313,768 | 54,974 | 47,880 | 34,587 | 81.72% | 86.76% | 90.07% | 0.884 | 0.458 |
| Naive Bayes |  | 2 | WNL | Sim | 316,304 | 53,330 | 49,524 | 32,051 | 81.92% | 86.46% | 90.80% | 0.886 | 0.457 |
| SVM | c=0.1 | 1 | WNL | Sim | 318,612 | 51,700 | 51,154 | 29,743 | 82.07% | 86.17% | 91.46% | 0.887 | 0.455 |
| SVM | c=20 | 1 | POS (rápido) | Sim | 311,624 | 55,683 | 47,171 | 36,731 | 81.41% | 86.85% | 89.46% | 0.881 | 0.453 |
| SVM | c=50 | 1 | Corretor | Sim | 313,139 | 54,654 | 48,200 | 35,216 | 81.51% | 86.66% | 89.89% | 0.882 | 0.452 |
| SVM | c=5 | 1 | Corretor | Sim | 313,282 | 54,492 | 48,362 | 35,073 | 81.51% | 86.63% | 89.93% | 0.882 | 0.451 |
| SVM | c=20 | 1 | Corretor | Sim | 312,542 | 54,907 | 47,947 | 35,813 | 81.44% | 86.70% | 89.72% | 0.882 | 0.451 |
| SVM | c=5 | 1 | POS (rápido) | Sim | 313,869 | 54,122 | 48,732 | 34,486 | 81.56% | 86.56% | 90.10% | 0.883 | 0.451 |
| SVM | c=50 | 1 | POS (rápido) | Sim | 313,144 | 54,521 | 48,333 | 35,211 | 81.48% | 86.63% | 89.89% | 0.882 | 0.451 |
| SVM | c=10 | 1 | Corretor | Sim | 313,043 | 54,526 | 48,328 | 35,312 | 81.46% | 86.63% | 89.86% | 0.882 | 0.450 |
| SVM | c=1 | 1 | POS (rápido) | Sim | 314,802 | 53,456 | 49,398 | 33,553 | 81.62% | 86.44% | 90.37% | 0.884 | 0.450 |
| SVM | c=10 | 1 | POS (rápido) | Sim | 312,433 | 54,842 | 48,012 | 35,922 | 81.40% | 86.68% | 89.69% | 0.882 | 0.450 |
| SVM | c=1 | 1 | Corretor | Sim | 314,754 | 53,459 | 49,395 | 33,601 | 81.61% | 86.44% | 90.35% | 0.884 | 0.450 |
| SVM | c=0.01 | 2 | Simples | Não | 325,679 | 46,206 | 56,648 | 22,676 | 82.42% | 85.18% | 93.49% | 0.891 | 0.448 |
| SVM | c=0.01 | 2 | POS (rápido) | Sim | 325,435 | 46,275 | 56,579 | 22,920 | 82.38% | 85.19% | 93.42% | 0.891 | 0.447 |
| SVM | c=0.01 | 2 | Corretor | Sim | 325,397 | 46,278 | 56,576 | 22,958 | 82.37% | 85.19% | 93.41% | 0.891 | 0.447 |
| SVM | c=0.01 | 2 | Simples | Sim | 325,755 | 45,977 | 56,877 | 22,600 | 82.39% | 85.14% | 93.51% | 0.891 | 0.447 |
| SVM | c=0.1 | 1 | Simples | Não | 320,340 | 48,998 | 53,856 | 28,015 | 81.86% | 85.61% | 91.96% | 0.887 | 0.442 |
| SVM | c=0.01 | 2 | POS (NB) | Sim | 326,971 | 44,528 | 58,326 | 21,384 | 82.33% | 84.86% | 93.86% | 0.891 | 0.441 |
| SVM | c=0.1 | 1 | Simples | Sim | 320,296 | 48,810 | 54,044 | 28,059 | 81.80% | 85.56% | 91.95% | 0.886 | 0.440 |
| SVM | c=0.1 | 1 | Corretor | Sim | 320,310 | 48,503 | 54,351 | 28,045 | 81.74% | 85.49% | 91.95% | 0.886 | 0.437 |
| SVM | c=0.1 | 1 | POS (NB) | Sim | 320,266 | 48,451 | 54,403 | 28,089 | 81.72% | 85.48% | 91.94% | 0.886 | 0.436 |
| SVM | c=0.1 | 1 | POS (rápido) | Sim | 320,070 | 48,558 | 54,296 | 28,285 | 81.70% | 85.50% | 91.88% | 0.886 | 0.436 |
| SVM | c=0.01 | 1 | WNL | Sim | 322,189 | 46,266 | 56,588 | 26,166 | 81.66% | 85.06% | 92.49% | 0.886 | 0.428 |
| SVM | c=0.001 | 2 | WNL | Sim | 326,047 | 41,602 | 61,252 | 22,308 | 81.48% | 84.18% | 93.60% | 0.886 | 0.410 |
| SVM | c=0.01 | 1 | POS (rápido) | Sim | 325,135 | 41,700 | 61,154 | 23,220 | 81.30% | 84.17% | 93.33% | 0.885 | 0.405 |
| SVM | c=0.01 | 1 | Corretor | Sim | 325,235 | 41,542 | 61,312 | 23,120 | 81.29% | 84.14% | 93.36% | 0.885 | 0.404 |
| SVM | c=0.01 | 1 | Simples | Não | 325,567 | 41,239 | 61,615 | 22,788 | 81.29% | 84.09% | 93.46% | 0.885 | 0.403 |
| SVM | c=0.01 | 1 | Simples | Sim | 325,517 | 41,164 | 61,690 | 22,838 | 81.27% | 84.07% | 93.44% | 0.885 | 0.402 |
| Naive Bayes |  | 2 | Simples | Não | 313,447 | 48,393 | 54,461 | 34,908 | 80.19% | 85.20% | 89.98% | 0.875 | 0.400 |
| Naive Bayes |  | 2 | Simples | Sim | 313,557 | 48,171 | 54,683 | 34,798 | 80.17% | 85.15% | 90.01% | 0.875 | 0.399 |
| Naive Bayes |  | 2 | Corretor | Sim | 314,708 | 47,330 | 55,524 | 33,647 | 80.24% | 85.00% | 90.34% | 0.876 | 0.397 |
| SVM | c=0.01 | 1 | POS (NB) | Sim | 326,371 | 39,795 | 63,059 | 21,984 | 81.15% | 83.81% | 93.69% | 0.885 | 0.395 |
| SVM | c=0.001 | 1 | WNL | Sim | 325,812 | 39,880 | 62,974 | 22,543 | 81.05% | 83.80% | 93.53% | 0.884 | 0.392 |
| SVM | c=0.001 | 2 | POS (rápido) | Sim | 329,287 | 36,774 | 66,080 | 19,068 | 81.13% | 83.29% | 94.53% | 0.886 | 0.386 |
| SVM | c=0.001 | 2 | Corretor | Sim | 329,230 | 36,708 | 66,146 | 19,125 | 81.10% | 83.27% | 94.51% | 0.885 | 0.385 |
| SVM | c=0.001 | 2 | Simples | Sim | 329,705 | 35,993 | 66,861 | 18,650 | 81.05% | 83.14% | 94.65% | 0.885 | 0.381 |
| SVM | c=0.001 | 2 | Simples | Não | 329,716 | 35,939 | 66,915 | 18,639 | 81.04% | 83.13% | 94.65% | 0.885 | 0.381 |
| SVM | c=0.001 | 2 | POS (NB) | Sim | 330,988 | 33,922 | 68,932 | 17,367 | 80.87% | 82.76% | 95.01% | 0.885 | 0.370 |
| SVM | c=0.001 | 1 | POS (rápido) | Sim | 329,072 | 35,049 | 67,805 | 19,283 | 80.70% | 82.92% | 94.46% | 0.883 | 0.368 |
| SVM | c=0.001 | 1 | Corretor | Sim | 329,161 | 34,903 | 67,951 | 19,194 | 80.69% | 82.89% | 94.49% | 0.883 | 0.367 |
| SVM | c=0.001 | 1 | Simples | Não | 329,674 | 34,324 | 68,530 | 18,681 | 80.67% | 82.79% | 94.64% | 0.883 | 0.365 |
| SVM | c=0.001 | 1 | Simples | Sim | 329,710 | 34,286 | 68,568 | 18,645 | 80.67% | 82.78% | 94.65% | 0.883 | 0.365 |
| Naive Bayes |  | 1 | POS (NB) | Sim | 335,167 | 29,519 | 73,335 | 13,188 | 80.82% | 82.05% | 96.21% | 0.886 | 0.357 |
| SVM | c=0.001 | 1 | POS (NB) | Sim | 330,388 | 32,129 | 70,725 | 17,967 | 80.34% | 82.37% | 94.84% | 0.882 | 0.348 |
| Boostexter | i=25 | 2 | WNL | Sim | 317,237 | 36,640 | 66,214 | 31,118 | 78.43% | 82.73% | 91.07% | 0.867 | 0.313 |
| Boostexter | i=25 | 2 | Corretor | Sim | 326,327 | 28,673 | 74,181 | 22,028 | 78.68% | 81.48% | 93.68% | 0.872 | 0.286 |
| Boostexter | i=25 | 2 | Simples | Sim | 326,787 | 28,145 | 74,709 | 21,568 | 78.66% | 81.39% | 93.81% | 0.872 | 0.284 |
| Naive Bayes |  | 1 | Simples | Não | 336,844 | 21,148 | 81,706 | 11,511 | 79.34% | 80.48% | 96.70% | 0.878 | 0.279 |
| Naive Bayes |  | 1 | WNL | Sim | 338,270 | 20,030 | 82,824 | 10,085 | 79.41% | 80.33% | 97.10% | 0.879 | 0.279 |
| Naive Bayes |  | 1 | Simples | Sim | 336,877 | 21,045 | 81,809 | 11,478 | 79.33% | 80.46% | 96.71% | 0.878 | 0.278 |
| Naive Bayes |  | 1 | Corretor | Sim | 338,428 | 18,447 | 84,407 | 9,927 | 79.09% | 80.04% | 97.15% | 0.878 | 0.261 |
| Boostexter | i=25 | 2 | POS (NB) | Sim | 330,404 | 23,543 | 79,311 | 17,951 | 78.44% | 80.64% | 94.85% | 0.872 | 0.257 |

# Apêndice B – Distribuição dos comentários por grupo

Como o objetivo da etapa II era de fazer a classificação de acordo com os grupos, a tabela abaixo contém a quantidade de mensagens por grupo.

* Corpus 1

| categoria | qtde |
| --- | --- |
| 9658 | 149,116 |
| 5603 | 81,483 |
| 6174 | 51,741 |
| 5604 | 42,845 |
| 16031 | 38,723 |
| 5598 | 32,465 |
| 5601 | 27,443 |
| 5602 | 24,804 |
| 7086 | 22,412 |
| 15605 | 19,781 |
| 5605 | 18,646 |
| 5606 | 15,647 |
| 10407 | 12,431 |
| 7085 | 11,050 |
| 9666 | 10,669 |
| 7084 | 9,307 |
| 16814 | 8,608 |
| 9654 | 8,234 |
| 9982 | 7,301 |
| 5599 | 6,264 |
| 10345 | 5,952 |
| 16029 | 5,496 |
| 6091 | 4,159 |
| 16030 | 3,991 |
| 8524 | 3,928 |
| 17671 | 3,854 |
| 46 | 3,458 |
| 16020 | 3,411 |
| 256 | 3,399 |
| 9101 | 3,026 |
| 9356 | 2,426 |
| 9097 | 2,346 |
| 5600 | 2,117 |
| 10041 | 1,699 |
| 105 | 1,426 |
| 10039 | 1,415 |
| 17082 | 1,281 |
| 144 | 1,251 |
| 16619 | 1,188 |
| 47 | 1,116 |
| 76 | 1,098 |
| 16726 | 1,092 |
| 44 | 1,039 |
| 9772 | 1,033 |
| 10040 | 1,021 |
| 94 | 790 |
| 15528 | 751 |
| 16306 | 599 |
| 109 | 566 |
| 39 | 558 |
| 17816 | 558 |
| 17815 | 442 |
| 15526 | 426 |
| 20 | 408 |
| 101 | 398 |
| 37 | 390 |
| 8 | 368 |
| 15637 | 360 |
| 16052 | 346 |
| 27 | 327 |
| 16108 | 306 |
| 15610 | 278 |
| 42 | 270 |
| 31 | 250 |
| 9662 | 220 |
| 17084 | 175 |
| 16307 | 148 |
| 18402 | 144 |
| 29 | 138 |
| 17083 | 121 |
| 17396 | 119 |
| 8610 | 84 |
| 17814 | 84 |
| 9099 | 81 |
| 148 | 78 |
| 17397 | 66 |
| 45 | 65 |
| 17812 | 64 |
| 15913 | 60 |
| 119 | 49 |
| 15524 | 48 |
| 9076 | 31 |
| 8334 | 26 |
| 9147 | 25 |
| 15530 | 22 |
| 142 | 15 |
| 16725 | 14 |
| 222 | 11 |
| 9141 | 9 |
| 5 | 9 |
| 16107 | 6 |
| 124 | 5 |
| 164 | 5 |
| 214 | 4 |
| 17856 | 3 |
| 178 | 2 |
| 15700 | 2 |
| 43 | 2 |
| 134 | 1 |
| 16634 | 1 |
| 16021 | 1 |
| 16022 | 1 |
| 146 | 1 |
| 147 | 1 |
| 16618 | 1 |

* Corpus 2

|  |  |  |
| --- | --- | --- |
| Categoria | Aprovados | Reprovados |
| Araguaia | 2.581 | 4.106 |
| Bom Dia Brasil | 128 | 416 |
| Caldeirão de Twittadas | 2.344 | 7.722 |
| Fantástico | 1.493 | 44.902 |
| G1 | 2.176 | 8.434 |
| G1 Carros | 56 | 1.051 |
| G1 Eleições | 889 | 2.566 |
| G1 Pop e Arte | 7 | 78 |
| G1 RJ | 31 | 240 |
| G1 SP | 176 | 3.804 |
| Globo News | 152 | 1.563 |
| Globo Repórter | 476 | 1.681 |
| Hipertensão | 4.597 | 21.920 |
| Inscrições BBB 11 | 26.361 | 86.523 |
| Jornal Hoje | 219 | 1.622 |
| Junto & Misturado | 1.414 | 13.288 |
| Passione | 19.110 | 97.774 |
| Ti-Ti-Ti | 39.781 | 49.575 |
| TV Garagem | 799 | 790 |
| Total | 102.790 | 348.055 |

# Apêndice C – Parâmetros de uso da API de linha de comando

usage: test\_datasets.py [-h] [--prob-distribution] --moderator

{baseline,random,svm,nb,boosting,me-iis,me-gis,me-cg}

[--feature {simple,pos-fast,pos-nb,wnl,spell}]

[--iter ITER] [--cross CROSS] [--ngrams NGRAMS]

[--limit LIMIT] [--strict-protection {0,1}]

[--svm SVM] [--specific-learn {0,1}]

dataset

Measure a moderator/features accuracy (and others metrics)

positional arguments:

dataset a dataset file where each line have a status (can be

-1 for rejected and 1 for approved), a category (any

string without comma) and a text, always in UTF-8. A

column separator is (tab character)

optional arguments:

-h, --help show this help message and exit

--prob-distribution Print probabilities distribution

--moderator {baseline,random,svm,nb,boosting,me-iis,me-gis,me-cg}

moderator implementation used to train dataset

--feature {simple,pos-fast,pos-nb,wnl,spell}

feature extractor method

--iter ITER number of iterations for boosting and maximum entropy

(me)

--cross CROSS cross validation

--ngrams NGRAMS use ngram together feature extractor

--limit LIMIT use only n random elements in dataset

--strict-protection {0,1}

Reject comments that have xss, link or e-mail (without

learning)

--svm SVM parameters only for SVM

--specific-learn {0,1}

Use specific learning per category