5
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 4 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 5.1 discute os critérios utilizados para avaliar a eficiência do nosso sistema. A Seção 5.2 descreve os corpora utilizados. A seção 5.3 aborda a metodologia para realizar o experimento utilizando os corpora. Por fim, a Seção 5.4 apresenta os resultados obtidos.

5.1.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.

5.1.1.Precisão

Mede a quantidade de comentários aprovados que o sistema reprovou

(reprovados incorretamente). É definida pela fórmula:

$$precisao = \frac{RC}{RC + RI}$$

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.

5.1.2.Recall

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

$$recall = \frac{RC}{RC + AI}$$

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.

5.1.3.Acurácia

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

$$acuracia = \frac{RC + AC}{RC + AC + RI + AI}$$

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%.

5.1.4.F-measure

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

$$F_\beta = 1 + \beta^2 \frac{precisao \cdot recall}{\beta^2 \cdot precisao + recall}$$

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 β. Nos casos em que β é definido como 1, esta métrica recebe o nome de F_1.

5.1.5.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:

$$MCC = \frac{RC \times AC - RI \times AI}{\sqrt{(RC + RI)(RC + AI)(AC + RI)(AC + AI)}}$$

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 F_1quanto o MCC são boas métricas para avaliar qual o melhor algoritmo a ser utilizado. Porém, neste trabalho optamos por 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 se encontra. Portanto, quando no restante do texto falarmos sobre eficiência do classificador, estaremos levando em consideração o valor da métrica MCC.

5.2.Corpora

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

5.2.1.Corpus globo-comments

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.

5.2.1.Corpus globo-twitter

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.

5.2.2.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 globo-twitter 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 globo-comments 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 globo-twitter será de confrontar os resultados obtidos no experimento com o corpus globo-comments.

- No corpus globo-comments 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.

5.3.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:

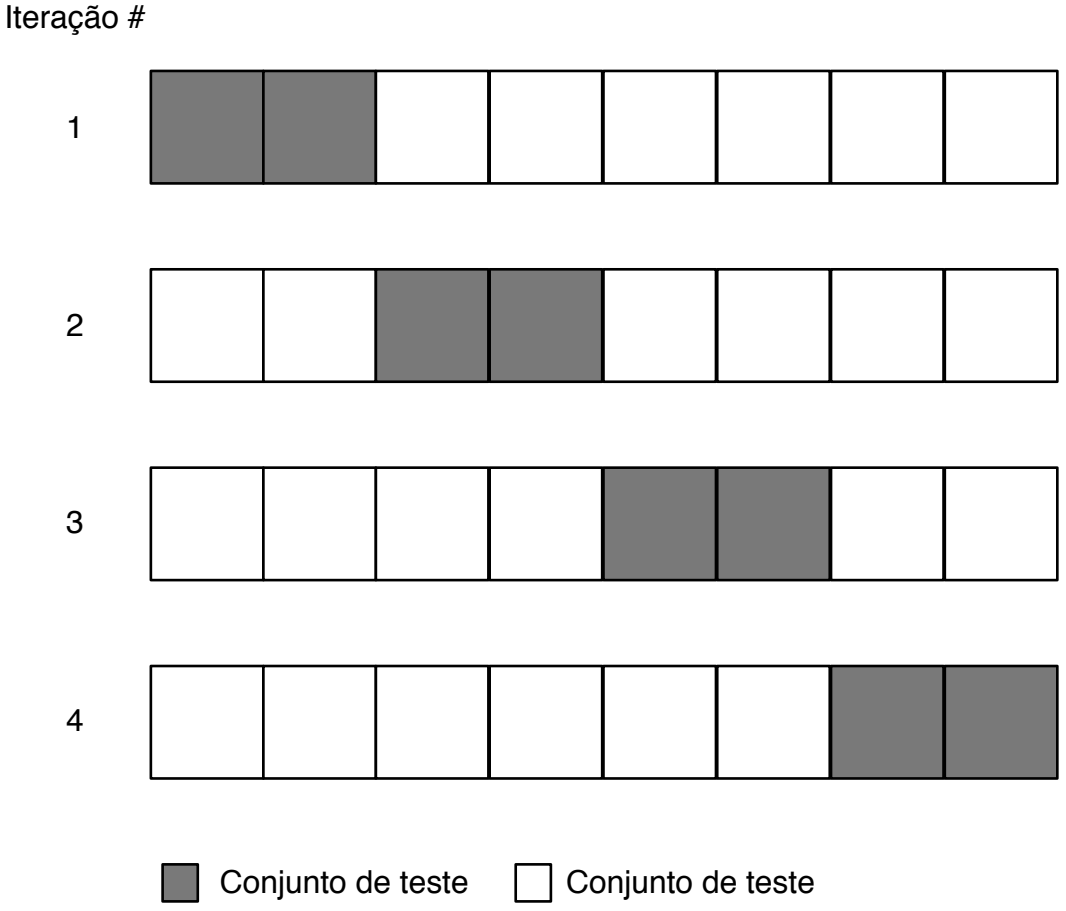


Figura 4 - Validação cruzada para 4 partições

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.

5.4.Resultados obtidos

Nesta seção discutimos 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 resposta mais adequada para o problema. A segunda etapa, com o algoritmo e o método de extração escolhidos, avalia-se 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. No que se segue, detalhamos cada uma das etapas do método proposto.

5.4.1.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. Em todos os testes foi utilizado validação cruzada com 10 partições.

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 globo-comments, o uso de palavras e números como atributos trouxe um melhor resultado, enquanto que no corpus globo-twitter, ao adicionarmos os caracteres de pontuação (WNL), o MCC aumentou.

Como o corpus globo-twitter 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 globo-twitter 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 globo-comments, adicioná-los apenas gerou ruído para o classificador.

Conforme citado na Seção 5.2.2, o corpus globo-comments é o corpus de referência, enquanto que o corpus globo-twitter 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..

5.4.2.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.

5.4.2.1.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 c o comentário que está sendo classificado, $prob_{global}(c)$ a probabilidade do comentário pelo moderador global e $prob_{grupo}(c)$ a probabilidade para o moderador específico (do grupo).

$$prob(c) = \begin{cases} prob_{global}(c), & \nexists\, prob_{grupo}(c) \\ prob_{global}(c), & |prob_{global}(c) - 0.5| > |prob_{grupo}(c) - 0.5| \\ prob_{grupo}(c), & |prob_{global}(c) - 0.5| \leq |prob_{grupo}(c) - 0.5| \end{cases}$$

5.4.2.2.Resultados da 2ª etapa

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	F_1	MCC
Corpus globo-comments	54,424	83,264	574,141	657,405	89,10%	34,64%	0.446	**0.412**
Corpus globo-twitter	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 globo-comments podemos verificar que o MCC saiu de 0.375 para 0.412, um aumento de quase 10%. A métrica F_1 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 globo-comments.

Em relação ao corpus globo-twitter, 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.

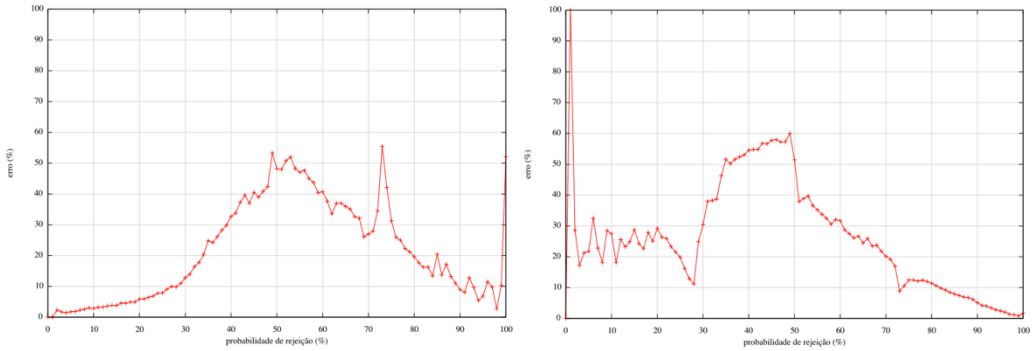
O grande problema da solução de utilizar classificadores de contexto é o consumo de recursos. Em relação a CPU, na pior hipótese, teríamos o consumo multiplicado por 2, pois para cada comentário seria necessário haver 2 classificadores (global e do grupo). Já em relação a memória, o consumo aumenta drasticamente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

5.4.3.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 3.1.3). 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.

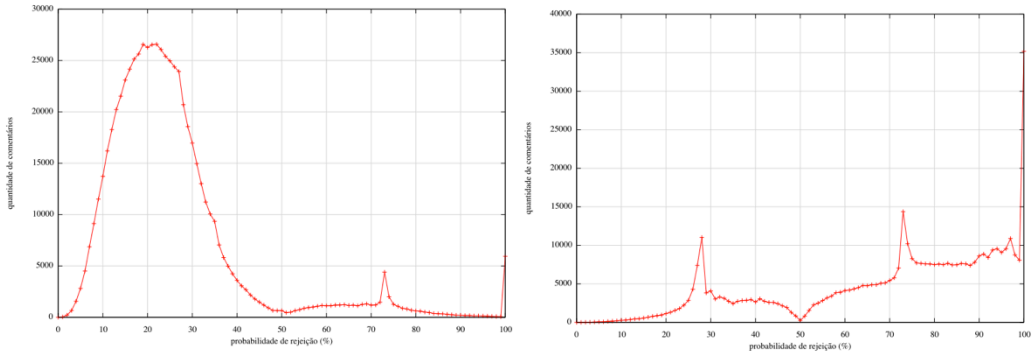


(I) (II)

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

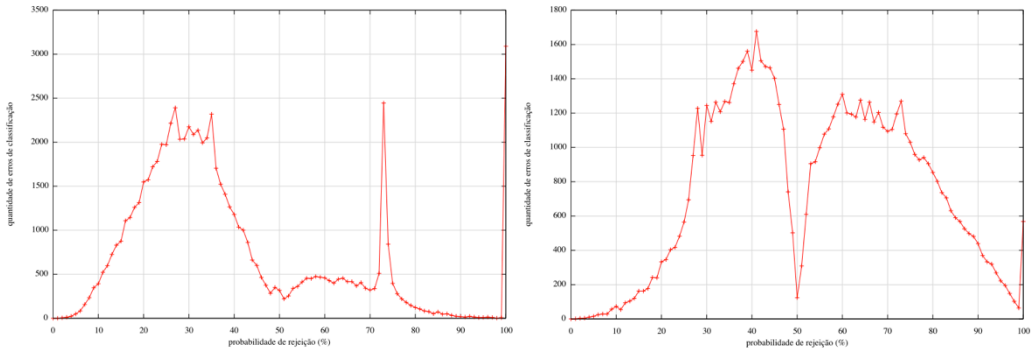
Observando os gráficos da figura 5 podemos verificar que a taxa de erro, relativo a probabilidade, é maior quando se está mais próximo da probabilidade de mudança de classe, que é de 50%. Do ponto do visto do SVM, são os comentários que estão mais próximos ao hiperplano que separa as duas classes. Porém, 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.

5.5.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 globo-comments. Um resultado análogo pode ser obtido utilizando o corpus globo-twitter.

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 globo-twitter 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 globo-twitter 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 globo-comments 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.