UNIVERSIDADE ESTADUAL DE MARINGÁ

PROGRAMA DE INICIAÇÃO CIENTÍFICA – PIC

DEPARTAMENTO DE INFORMÁTICA

ORIENTADOR (A): Drª Profª Valéria Delisandra Feltrim

ACADÊMICO (s): Alessandra Harumi Iriguti

CRIAÇÃO DE CLASSIFICADORES RETÓRICOS PARA RESUMOS CIENTÍFICOS DA PUBMED

Maringá, de de .

UNIVERSIDADE ESTADUAL DE MARINGÁ

PROGRAMA DE INICIAÇÃO CIENTÍFICA – PIC

DEPARTAMENTO DE INFORMÁTICA

ORIENTADOR(A): Drª Profª Valéria Delisandra Feltrim

ACADÊMICO(s): Alessandra Harumi Iriguti

CRIAÇÃO DE CLASSIFICADORES RETÓRICOS PARA RESUMOS CIENTÍFICOS DA PUBMED

Relatório contendo os resultados finais do projeto de iniciação científica vinculado ao Programa PIC-UEM.

Maringá, de de .

Resumo (máx. 150 palavras)

Classificar as estruturas retóricas auxiliam no desenvolvimento e na compreensão de textos. As categorias retóricas variam de acordo com o modelo retórico adotado. Para os corpora em inglês, foram utilizadas as categorias *conclusions*, *methods*, *objectives* e *results* e, para os corpora em português, contexto, conclusão, lacuna, metodologia, estrutura, propósito e resultado. Visando avaliar o desempenho de redes neurais profundas na tarefa de classificação retórica em comparação a outros classificadores supervisionados, tem-se como objetivo implementar classificadores sentenciais para resumos extraídos da base PubMed e de resumos, em português, coletados manualmente de textos científicos de computação. Os algoritmos de aprendizagem de máquina utilizados são SVM, *Nearest Neighbors*, *Naive Bayes* e *Decision Trees*, combinados com diferentes atributos como *tf-idf*, n-gramas, seleção de *features*, posição e a estimativa de validação cruzada. Devido à limitação de memória não foi possível rodar todas as combinações. No geral, os fatores que mostraram melhores performances foram: unigrama ou bigrama, aumentar atributos, selecionar *features*, aumento de sentenças (para os corpora em inglês) e utilizar *cross-validation*. O algoritmo que apresentou melhor performance foi o SVM.

Sumário

1. Introdução

As estruturas discursivas são os padrões vistos em textos multi-sentenças (multi-clausal). Reconhecer esses padrões em termos dos elementos que os compõem é essencial para derivar e interpretar corretamente a informação no texto. Os elementos podem ser funções, cada uma realizada por uma ou mais cláusulas, em que a função servida pode ser com respeito ao discurso como um todo ou algum outro segmento do discurso. (WEBBER; EGG; KORDONI, 2011).

Os elementos de um discurso e sua posição no discurso são chamados de estrutura retórica. Uma estrutura retórica é formada por um conjunto de relações retóricas e são essas relações que definem como seu conteúdo está relacionado e como cada pedaço de conteúdo de texto, também chamado de proposição, contribui para satisfazer as intenções do autor. (ROMEIRO, 2016). A estrutura retórica pode ser representada de forma mais geral (para qualquer discurso) ou mais específica (para um gênero textual em particular).

Para estruturas retóricas especificamente para gênero científicos, tem-se, por exemplo, uma pesquisa experimental, o *Writing Up Research*, feito por Weissberg e Buker (1990). O objetivo deste é auxiliar escritores iniciantes na escrita destes textos científicos. São apresentados modelos estruturais das partes que compõem um texto científico, inclusive, de resumos.

São implementados classificadores treinados por algoritmos de aprendizagem de máquina (AM) para detectar automaticamente as estruturas retóricas. Segundo Russel e Norvig (1995, p. 1, apud LORENA e CARVALHO, 2003), "Por classificação entende-se o processo de atribuir, a uma determinada informação, o rótulo da classe à qual ela pertence". Dessa forma, utiliza-se técnicas de AM para indução do classificador. (LORENA; CARVALHO, 2003). Para essa indução, usa-se um indutor ou algoritmo de aprendizado. Dentre esses algoritmos de aprendizado, um dos mais utilizados são os supervisionados, por exemplo, *Support Vector Machine*, *Naive Bayes*, *Nearest Neighbors* e *Decision Trees*. Neles, o indutor recebe um conjunto de exemplos de treinamento para os quais o rótulo (*label*) da classe associada é conhecido. (REZENDE, 2003). Esses algoritmos utilizam diferentes paradigmas de aprendizado como: simbólico, estatísticos, baseados em exemplos, conexionista e evolutivo. Os algoritmos a serem utilizados se enquadram apenas nos três primeiros. Os simbólicos utilizam símbolos gerados a partir da análise de exemplos e contraexemplos, como é o caso do *Decision Trees*. Já os estatísticos utilizam modelos estatísticos para encontrar uma aproximação do conceito induzido que seja aceitável, por exemplo, *Support Vector Machine* e *Naive Bayes*. Por

fim, os baseados em exemplos necessitam manter os exemplos em memória para classificar as sentenças através de suas similaridades. O *Nearest Neighbors* é um dos algoritmos mais conhecidos deste paradigma.

2. Objetivos

O objetivo deste trabalho de iniciação científica é implementar classificadores retóricos sentenciais para resumos extraídos da base MEDLINE/PubMed e um conjunto de textos em português coletados manualmente pela orientadora deste projeto, visando utilizá-los como base de comparação induzidos por redes neurais profundas. Tais classificadores *baselines* utilizarão algoritmos conhecidos de aprendizagem supervisionada, tais como SVM, Nearest Neighbors, Naive Bayes e Decision Trees, e atributos que possam ser extraídos da superfície das sentenças dos resumos, tais como n-gramas, *tf-idf* e posição da palavra.

3. Desenvolvimento (Materiais e Métodos)

Para os testes dos classificadores, foram utilizados cinco corpora de resumos. Dois foram extraídos da base MEDLINE/PubMed e se encontram em inglês. Os outros três se encontram na língua portuguesa (do Brasil) e foram retirados de uma coletânea de textos. Foram testados os algoritmos: Suport Vector Machine, Nearest Neighbors, Naive Bayes e Decision Trees. Os algoritmos utilizados foram importados da biblioteca *scikit-learn*, na linguagem Python. Para cada um destes algoritmos de aprendizagem supervisionada foram testados sem e com *Feature Selection*, com valores de *n-grams* de 1, 2 (para os corpora em inglês) e de 1, 2 e 3 (para os corpora em português) e utilizando *cross-validation*.

		dev	***data***
	Nº de resumos	834	93909
Labels	*conclusions*	15432	170168
	methods	26241	255730
	objectives	13325	132517
	results	35366	363734
	Total sentenças	**90364**	**922149**

Tabela 1 – Quantidade de resumos e sentenças dos corpora dev e data

Os dois primeiros corpora, retirados da base MEDLINE/PubMed, estão em inglês e utilizam as categorias retóricas: *conclusions*, *methods*, *objectives* e *results*. Essas categorias são quatro de cinco da *National Library of Medicine* (NLM), em que a quinta seria *background*. O primeiro corpus, chamado de *dev*, contém 8341 resumos, 15432 sentenças *conclusions*, 26241 *methods*, 13325 *objectives* e 35366 *results*, totalizando 90364 sentenças. O segundo, chamado de *data*, contém 93909 resumos e 922149 sentenças, sendo 170168 sentenças de *conclusions*, 255730 *methods*, 132517 *objectives* e 363734 *results*. Tais dados se encontram na tabela 1.

Os outros foram coletados manualmente pela orientadora Drª Valéria Feltrim, durante seu trabalho de doutorado de teses e dissertações de computação. Estes corpora são chamados de *366*, *466* e *832*, estão em português-BR e utilizam as *labels*: B, C, G, M, O, P e R. Tais letras representam as categorias retóricas: contexto, conclusão, lacuna, metodologia, estrutura, propósito e resultado, respectivamente. Como mostra a tabela 2, o corpus 366 possui 52 resumos, 77 sentenças B, 20 sentenças C, 36 sentenças G, 45 sentenças M, 6 sentenças O, 65 sentenças P e 117 sentenças R. Já o 466 possui 179 sentenças B, 20 sentenças C, 36 sentenças G, 59 sentenças M, 1 sentença O, 68 sentenças P, 103 sentenças R e, assim como o 366, também é composto de 52 resumos. Por fim, o 832 é a junção do corpus 366 com o 466. Consequentemente, possui 104 resumos, 256 sentenças B, 40 sentenças C, 72 sentenças G, 104 sentenças M, 7 sentenças O, 133 sentenças P e 220 sentenças R.

		366	*466*	*832*
	Nº de resumos	52	52	104
Labels	B	77	179	256
	C	20	20	40
	G	36	36	72
	M	45	59	104
	O	6	1	7
	P	65	68	133
	R	117	103	220
	Total sentenças	**366**	**466**	**832**

Tabela 2 – Quantidade de resumos e sentenças dos corpora 366, 466 e 832

Nos corpora, cada sentença se encontra na forma ["$label$","$sentença$"]. Na *label* está a classe retórica da sentença e uma sentença começa depois de um ponto final (.) ou no começo do resumo e termina em um ponto final. Um conjunto dessas sentenças, nesta forma, forma um

resumo ([["$label_1$","$sentença_1$"], ..., ["$label_n$","$sentença_n$"]]) e um conjunto de resumos forma o corpus ([[["$label_1$","$sentença_1$"], ..., ["$label_n$","$sentença_n$"]], ..., [["$label_1$","$sentença_1$"], ..., ["$label_n$","$sentença_n$"]]]).

O primeiro algoritmo testado foi o *Suport Vector Machine*. Nesse, foi utilizado o *LinearSVC* que é a implementação do SVM da biblioteca *scikit-learn*. Sua implementação é baseada no *LibSVM*, contendo termos de *LibLinear*. Os parâmetros foram: $dual = False$ e $tol = 1e - 3$. O segundo foi *KNeighborsClassifier*. Os parâmetros para o *Nearest Neighbors* foram: $n\_neighbors = 15$ e $weight = 'uniform'$. Outro algoritmo, o *MultinomialNB* foi usado para testar o *Naive Bayes*, que não possui parâmetros. Por fim, para o *Decision Trees*, foi usado o *DecisionTreeClassifier* com o parâmetro $random\_state = 0$. Todos esses algoritmos foram importados da biblioteca *scikit-learn*.

Para cada algoritmo, foi aplicado o estimador de validação cruzada com valor 10 (*10-fold cross-validation*). Além disso, foram usados *tf-idf* para unigramas, bigramas e trigramas. A tabela 3 mostra a quantidade de *features* para n-gramas de cada corpus.

	dev	***data***	***366***	***466***	***832***
***ngram*=1**	45938	136645	2070	2517	3582
***ngram*=2**	629553	3219867	8011	9874	15805
***ngram*=3**	1836892	12454420	15340	18774	31720

Tabela 3 - Quantidade de features *de cada corpus para cada valor de n_grama*

Dentre essas características geradas, ou eram todas usadas ou eram selecionadas as *k features* com melhores pontuações (*SelectKBest* dos maiores *tf-idf*). Para essa pontuação, o teste de qui-quadrado (chi-squared, chi2) foi aplicado. Este teste mede a dependência entre as variáveis estocásticas (especificamente, entre resultados experimentais e a distribuição esperada). Assim, as *features* que são mais prováveis de serem independentes da classe são eliminadas, já que são irrelevantes para a classificação. Há, para cada valor de n-grama de cada corpus, uma variação de valores para k, já que o número de *features* são variados. Os valores de k começavam em um número igual ao total de características (que seria, sem seleção), que passava para um número próximo à quantidade de *features*, e ia reduzindo (aproximadamente metade) até chegar em um valor baixo (50, em alguns casos). Por exemplo, para o corpus *366* com $ngram = 3$, foram usados: sem seleção, $k = 10000$, $k = 5000$ e $k = 2500$, $k = 1000$, $k = 500$, $k = 250$, $k = 100$ e $k = 50$.

Para cada corpus, cada algoritmo foi aplicado duas vezes. Na primeira, foram usadas as seguintes características: *tf-idf* da palavra, *tf-idf* da palavra anterior, *tf-idf* da palavra posterior e a posição da palavra. Já na segunda vez, apenas as características *tf-idf* da palavra e posição da palavra foram utilizadas. Em ambas, para cada valor de n-grama, uma vez era sem seleção de *feature* e outras vezes aplicando chi2 na seleção para vários valores k. Além disso, realizou-se, também, treino com o corpus 466 e teste no 366. Sob os mesmos parâmetros e mesmas estratégias, com exceção de que não foi feito validação cruzada.

4. Resultados e Discussão

Nos corpora *dev* e *data*, não foi possível aplicar os algoritmos Nearest Neighbors e Decision Trees, porque o computador em que foi realizado os testes não tinha memória suficiente. Por esse mesmo motivo, não foi usado trigramas para o corpus *data*.

A notação utilizada aqui para os algoritmos é X_1 para o algoritmo X que utiliza *tf-idf* da palavra, *tf-idf* da palavra anterior, *tf-idf* da palavra posterior e a posição da palavra como *features*; e X_2 é o algoritmo X que tem apenas *tf-idf* da palavra e a posição.

4.1. Resultados do corpus *dev*

Na tabela 4, é mostrado a média do *f1-score* dos algoritmos SVM (*Support Vector Machine*) e NB (*Naive Bayes*) para o corpus *dev*.

O algoritmo SVM_1 obteve um desempenho de 0,92 para n-grama igual a 2 e igual a 3. A partir dele, pode-se observar uma melhora de performance ao aumentar o valor de n-grama, já que, com n-grama igual a 1, os valores atingidos foram 0,90 e 0,91. É possível observar também que, com as características utilizadas, o algoritmo se manteve regular, sem muitas oscilações. Por outro lado, a redução de *features* fez com que o SVM_2 obtivesse valores menores e oscilasse entre 0,87 e 0,90.

O NB_1 obteve resultados menores que o SVM_2. Seu melhor resultado foi 0,88 e o pior foi 0,53. O melhor do NB_2 foi 0,84 e o pior foi o menor de todos da tabela com 0,45. Com isso, pode ser afirmado que, para este corpus, o Naive Bayes com o aumento do valor de n-grama não é efetivo. Além disso, não usar seleção de atributos também não é vantajoso.

	k	SVM_1	SVM_2	NB_1	NB_2
***ngram* = 1**	sem	0,90	0,87	0,83	0,79
	10000	0,90	0,88	0,88	0,84
	5000	0,91	0,88	0,88	0,84
	2500	0,91	0,88	0,88	0,84
***ngram* = 2**	sem	0,92	0,89	0,58	0,52
	100000	0,92	0,90	0,75	0,72
	50000	0,92	0,90	0,83	0,79
	25000	0,92	0,90	0,87	0,83
	10000	0,92	0,89	0,88	0,84
	5000	0,92	0,89	0,88	0,83
***ngram* = 3**	sem	0,92	0,90	0,53	0,45
	200000	0,92	0,90	0,62	0,57
	100000	0,92	0,90	0,71	0,66
	50000	0,92	0,90	0,79	0,74
	25000	0,92	0,90	0,84	0,78
	10000	0,92	0,89	0,87	0,80
	5000	0,92	0,88	0,87	0,79

Tabela 4 – Média de f1-score *dos algoritmos para o corpus* dev

O maior valor é 0,92, obtido pelo algoritmo SVM_1 e o menor é 0,45 com NB_2. As tabelas 5 e 6 mostram, respectivamente, um relatório e a matriz de confusão destes dois algoritmos que resultaram nesses dois valores. No caso do SVM_1, foi escolhido qualquer um em que o n-grama seja estritamente maior que 1.

	SVM_1 (ngram=3, k=10000)			NB_2 (ngram=3, sem chi2)		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
conclusions	0,91	0,87	0,89	0,25	0,01	0,03
methods	0,93	0,93	0,93	0,72	0,24	0,36
objectives	0,97	0,95	0,96	1,00	0,41	0,58
results	0,90	0,93	0,92	0,47	1,00	0,64
avg/total	**0,92**	**0,92**	**0,92**	**0,58**	**0,52**	**0,45**

Tabela 5 – Relatório de precision, recall *e* f1-score *para cada* label

Pela tabela 5, observa-se que a precisão do SVM_1 que as sentenças de objetivos foram as que obtiveram melhor resultado, tanto em *precision* quanto em *recall*. Consequentemente, seu *f1-score* também foi melhor com 96%. Como é possível destacar na terceira coluna da tabela 6, os números que não estão na diagonal principal são relativamente baixos. Se somarmos

os valores em uma coluna, nós obtemos a quantidade que o algoritmo predisse sobre uma classe. É interessante observar que na terceira coluna (referente à *objectives*), é a menor soma, ou seja, o algoritmo classificou menos sentenças como objetivos e, ainda assim, foi o que obteve melhor performance.

SVM_1 (ngram=3, k=10000)					NB_2 (ngram=3, sem chi2)			
13463	39	78	1852	*conclusions*	205	6	1	15220
103	24409	203	1526	*methods*	11	6369	0	19861
86	498	12616	125	*objectives*	606	2403	5483	4833
1119	1176	64	33007	*results*	8	12	0	35346

Tabela 6 – Matriz de confusão de SVM_1 (ngram=3, k=10000) e NB_2 (ngram=3, sem chi2)

Já para o NB_2, observa-se algo menos regular. A classificação de conclusão teve a pior performance de apenas 3%, como pode ser visto na tabela 5. Em contraste com SVM_1, na tabela 6, vê-se que as sentenças *conclusions* foram as que o algoritmo menos classificou. O NB_2 mostrou-se melhor na classificação de sentenças de *results*, que foi o que o algoritmo mais classificou, quase 84% das sentenças do corpus foram classificadas como resultados. Exceto para *methods*, o NB_2 classificou a maioria das outras sentenças como sendo *results*.

4.2. Resultados do corpus *data*

A tabela 7 apresenta a média dos *f1-scores* dos algoritmos SVM e NB para o corpus *data*. Nela, observamos que o SVM ainda se manteve regular e, o NB, oscilante.

	K	SVM_1	SVM_2	NB_1	NB_2
$ngram = 1$	sem	0,92	0,89	0,87	0,84
	100000	0,92	0,89	0,88	0,85
	50000	0,92	0,90	0,88	0,85
	25000	0,92	0,90	0,89	0,85
	10000	0,92	0,90	0,89	0,69
$ngram = 2$	sem	0,94	0,91	0,69	0,69
	1000000	0,93	0,91	0,81	0,80
	500000	0,93	0,91	0,87	0,85
	250000	0,93	0,91	0,89	0,87
	100000	0,93	0,91	0,90	0,87
	50000	0,93	0,91	0,90	0,87

Tabela 7 – Média de f1-score *dos algoritmos para o corpus* data

Houve uma melhora na performance de ambos algoritmos neste corpus. O SVM_1 variou de 92% a 94% (melhor resultado), enquanto o SVM_2 foi de 89% a 91%. O NB_1 teve como menor valor 0,69 e, 0,90 como maior. Por fim, o NB_2 oscilou entre 0,69 e 0,87. O maior valor da tabela 7 foi o 0,94 do SVM_1, e, o menor, 0,69 do NB_1 e NB_2.

Para o SVM, observa-se que o aumento de n-gramas ajuda na performance, ao contrário da seleção de *features*, que não apresenta melhora, assim como ocorreu no corpus anterior. Já para o NB, o aumento de n-gramas e o uso de seleção de atributos mostrou uma melhora na performance, diferente de quando usado no corpus *dev*. Em ambos algoritmos, maior número de atributos (*tf-idf* da palavra, da anterior e da próxima e posição) colaboram para uma performance melhor.

	SVM_1 (ngram=2, sem chi2)			**NB_1 (ngram=2, sem chi2)**		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
conclusions	0,92	0,91	0,92	0,92	0,31	0,46
methods	0,94	0,94	0,94	0,82	0,66	0,73
objectives	0,98	0,97	0,97	1,00	0,56	0,72
results	0,92	0,93	0,93	0,61	0,99	0,76
avg/total	**0,94**	**0,94**	**0,94**	**0,78**	**0,71**	**0,69**

Tabela 8 – Relatório de precision, recall *e* f1-score *para cada* label

Nas tabelas 8 e 9, são apresentados resultados dos algoritmos que obtiveram a melhor média de *f1-score*, SVM_1, e a pior média, foi escolhido o NB_1. Assim como aconteceu para o corpus *dev*, a menor soma de predições é de *objectives* (como pode ser visto na tabela 9), mas foi o que teve maior pontuação, de 97% (tabela 8). Em geral, o SVM se mostrou eficiente e regular, mantendo sua performance acima de 90%. O maior valor do NB, assim como no corpus *dev*, foi melhor em *results* com uma porcentagem maior, de 76%. A menor eficiência também está para sentenças de conclusão, porém agora o valor aumentou de 3% para 46%.

SVM_1 (ngram=2, sem chi2)					**NB_1 (ngram=2, sem chi2)**			
155103	209	46	14810	*conclusions*	52710	10	2	117446
487	239774	1928	13541	*methods*	278	167636	2	87814
218	3827	128005	467	*objectives*	2555	33879	74074	22009
12336	11446	505	339447	*results*	1465	1876	0	360393

Tabela 9 – Matriz de confusão de SVM_1 (ngram=2, sem chi2) e NB_1 (ngram=2, sem chi2)

Em ambos corpora, os dois algoritmos mostraram menor eficiência em classificar sentenças *conclusions* e uma tendência de classificar as sentenças como *results*. No *dev*, o SVM classificou 40% como resultados e o NB quase 84%. Enquanto que para o*data*, o SVM classificou quase 39,9% das sentenças como *results* e o NB 63,7%.

4.3. Resultados do corpus *366*

A tabela 10 mostra os *f1-scores* dos algoritmos SVM, NN (Nearest Neighbors), NB e DT (Decision Trees) para n-grama de 1 a 3, com e sem seleção de *features*.

ngram	k	SVM_1	SVM_2	NN_1	NN_2	NB_1	NB_2	DT_1	DT_2
1	sem	0,47	0,50	0,38	0,34	0,30	0,33	0,42	0,42
	1000	0,55	0,55	0,31	0,32	0,34	0,34	0,44	0,48
	500	0,57	0,56	0,29	0,34	0,33	0,34	0,46	0,46
2	sem	0,42	0,44	0,37	0,33	0,26	0,27	0,42	0,44
	4000	0,48	0,50	0,26	0,26	0,24	0,26	0,44	0,47
	2000	0,57	0,51	0,28	0,31	0,26	0,26	0,46	0,50
	1000	0,53	0,54	0,33	0,28	0,23	0,23	0,45	0,50
	500	0,49	0,49	0,34	0,31	0,19	0,21	0,52	0,55
	250	0,48	0,45	0,36	0,33	0,19	0,19	0,49	0,54
	100	0,43	0,43	0,36	0,36	0,16	0,16	0,42	0,42
3	sem	0,39	0,41	0,37	0,28	0,22	0,20	0,40	0,43
	10000	0,43	0,42	0,25	0,25	0,19	0,18	0,43	0,48
	5000	0,48	0,47	0,31	0,26	0,21	0,20	0,47	0,47
	2500	0,50	0,49	0,33	0,27	0,20	0,20	0,46	0,48
	1000	0,46	0,46	0,34	0,32	0,17	0,17	0,45	0,54
	500	0,44	0,43	0,33	0,32	0,16	0,19	0,48	0,52
	250	0,40	0,40	0,35	0,33	0,15	0,17	0,47	0,45
	100	0,41	0,40	0,40	0,37	0,16	0,16	0,45	0,44
	50	0,41	0,40	0,40	0,38	0,16	0,16	0,43	0,44

Tabela 10 – Média de f1-score *dos algoritmos para o corpus* 366

Para o SVM (com ambas combinações de atributos), o aumento de n-grama não apresenta melhoras significativas na performance. Para unigrama selecionar os 500 melhores

(aproximadamente ¼ de *features*) mostrou melhor resultado 57%. Essa mesma pontuação foi atingida com bigrama para $k = 4000$ (aproximadamente ¼, também). Valores de k altos (superior à metade do total de *features*) mostraram resultados inferiores a 0,57, porém reduzir mais do que ¼ também não oferece uma performance melhor. É interessante observar, também, que sem seleção de *features*, o SVM_2 se saiu melhor que SVM_1.

O NN_1 foi um algoritmo que oscilou entre 0,25 e 0,40, enquanto o NN_2 entre 0,25 e 0,38. A variação de n-gramas não mostrou grandes influências na performance para ambos NN. Já a variação de quantidade de *features* selecionadas pareceu influenciar os resultados. Não fazer seleção e usar seleção com valor de *k* baixos (100, 50) foram os fatores que levaram aos melhores resultados do NN (no caso de NN_1, valores próximos ou iguais a 40% e, no caso de NN_2, valores próximos ou iguais a 38%).

Assim como aconteceu com o corpus *dev*, aumento de n-gramas não favorece a performance do NB. Para n-grama igual a 1, o algoritmo obteve resultados maiores ou iguais a 30%, porém, para n-grama igual a 2 e 3, os resultados foram menores que 30%, com 15% sendo o menor de todos, inclusive da tabela 10. Analisando então as linhas de n-grama=1, o NB_2, que utiliza menos atributos, se demonstrou mais constante quanto à variação de k. Sem seleção, $k = 1000$ e $k = 500$, o NB_2 pontou com 33%, 34% e 34%, respectivamente, enquanto o NB_1, 30%, 34% e 33%.

Nas duas últimas colunas da tabela 10, observa-se que a maioria dos resultados do DT_2 foi melhor que do DT_1 e que os melhores resultados do DT se encontram para n-grama igual a 2. A melhor performance do DT_2 foi igual a 55% e do DT_1 igual a 52%. É possível observar, também, que é a variação da quantidade de atributos selecionados que demonstrou ter uma influência maior nesse algoritmo, assim como aconteceu com o NB.

No geral, para o corpus *366*, o SVM se saiu melhor, com SVM_1 melhor que SVM_2. O DT ficou em segundo, com o DT_2 tendo performance melhor que DT_1. Em seguida vem o NN, com NN_1 melhor. E, por último, o NB, em que o NB_2 se demonstrou mais constante.

Selecionando dois resultados, o melhor, 57% do SVM_1, e o pior, 15% do NB_1, são apresentados, nas tabelas 11 e 12, o desempenho para cada classe e a matriz de confusão, respectivamente. Os parâmetros para o SVM_1 escolhido foram n-grama=1 e k=500 e, para o NB_1, n-grama=3 e k=250.

	SVM_1 (ngram=1, k=500)			NB_1 (ngram=3, k=250)		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
B	0,64	0,77	0,70	0,00	0,00	0,00
C	1,00	0,05	0,10	0,00	0,00	0,00
G	0,83	0,14	0,24	0,00	0,00	0,00
M	0,77	0,22	0,34	0,00	0,00	0,00
O	0,00	0,00	0,00	0,00	0,00	0,00
P	0,84	0,75	0,80	0,00	0,00	0,00
R	0,54	0,90	0,67	0,32	1,00	0,48
avg/total	**0,69**	**0,63**	**0,57**	**0,10**	**0,32**	**0,15**

Tabela 11 – Relatório de precision, recall *e* f1-score *para cada* label

Nas tabelas 11 e 12, é relevante observar que nenhum dos dois algoritmos conseguiram fazer a predição correta de *O (estrutura)*, eles não classificaram sentença alguma como sendo estrutura. Das 152 execuções de algoritmos, apenas em nove não ocorreu o valor zero e sete desses nove vieram do DT_2.

O SVM foi melhor na predição de sentenças *P*, com *f1-score* de 80%. Apesar de ter acertado 105 sentenças de resultado (*R*), o algoritmo apresentou uma certa tendência em classificar as sentenças como sendo resultado. Classificou 91 sentenças como *R* que não eram. Além de estrutura, outros dois que o SVM classificou pouco foram *C* e *G* – conclusão e lacuna. Como *C* foi apenas uma, corretamente, por isso sua precisão na tabela 11 aparece 100%. E como *G*, apenas 6, acertando 5, porém existiam mais 31 sentenças de lacuna, que foram 20 classificadas como *B* (contexto) e 11 como *R* (resultado).

SVM_1 (ngram=1, k=500)								NB_1 (ngram=3, k=250)						
59	0	1	0	0	4	13	*B*	0	0	0	0	0	0	77
1	1	0	0	0	0	18	*C*	0	0	0	0	0	0	20
20	0	5	0	0	0	11	*G*	0	0	0	0	0	0	36
2	0	0	10	0	1	32	*M*	0	0	0	0	0	0	45
0	0	0	0	0	1	5	*O*	0	0	0	0	0	0	6
4	0	0	0	0	49	12	*P*	0	0	0	0	0	0	65
6	0	0	3	0	3	105	*R*	0	0	0	0	0	0	117

Tabela 12 – Matriz de confusão de SVM_1 (ngram=1, k=500) e NB_1 (ngram=3, k=250)

Já, a baixa performance do NB neste corpus, se deve ao fato de que o algoritmo classificou todas as 366 sentenças como sendo *R*. Por isso, todas as colunas da tabela 12, com exceção da última, estão com todos os campos preenchidos com o valor zero. O desempenho de 48% para resultados se deve apenas porque 117 das 366 sentenças eram resultados.

4.4. Resultados do corpus *466*

A tabela 13 mostra a média de *f1-score* de SVM, NN, NB e DT para o corpus *466*. Assim como no corpus anterior, o aumento de n-grama não melhora de maneira significativa para o SVM. Sob os mesmos valores de k, o desempenho dele com trigrama é pior para este corpus maior. Da mesma forma que nos corpus anteriores, o SVM com mais atributos apresenta melhores resultados do que o SVM_2. O maior valor do SVM, inclusive da tabela 13, é 57%, igualmente ao *366*, com n-grama igual a 1 e $k = 500$. Comparando ao corpus *366*, a performance, no geral, foi semelhante, porém levemente pior.

ngram	k	SVM_1	SVM_2	NN_1	NN_2	NB_1	NB_2	DT_1	DT_2
1	sem	0,47	0,47	0,34	0,33	0,38	0,37	0,42	0,43
	2000	0,48	0,48	0,28	0,28	0,39	0,37	0,42	0,41
	1000	0,54	0,54	0,27	0,28	0,41	0,39	0,41	0,43
	500	0,57	0,53	0,26	0,26	0,40	0,35	0,48	0,49
2	sem	0,43	0,45	0,33	0,32	0,36	0,34	0,43	0,39
	5000	0,43	0,46	0,27	0,28	0,37	0,34	0,43	0,42
	2500	0,48	0,49	0,27	0,27	0,36	0,34	0,45	0,49
	1000	0,52	0,48	0,26	0,26	0,34	0,33	0,48	0,49
	500	0,46	0,45	0,26	0,26	0,32	0,29	0,49	0,49
	250	0,42	0,42	0,26	0,26	0,30	0,25	0,45	0,44
3	sem	0,37	0,40	0,32	0,33	0,36	0,33	0,42	0,38
	10000	0,34	0,38	0,26	0,27	0,35	0,34	0,41	0,42
	5000	0,40	0,42	0,27	0,26	0,35	0,33	0,43	0,44
	2500	0,49	0,46	0,26	0,26	0,33	0,32	0,47	0,50
	1000	0,45	0,42	0,26	0,27	0,32	0,31	0,49	0,50
	500	0,38	0,38	0,26	0,25	0,31	0,29	0,46	0,44

Tabela 13 – Média de f1-score *dos algoritmos para o corpus* 466

O NN se mostrou indiferente quanto à quantidade de atributos e à variação de n-grama. Tanto o NN_1 quanto o NN_2 apresentaram valores semelhantes. Além disso, o uso de seleção de *features* prejudicou o desempenho do algoritmo. Sem seleção, variou de 32% a 34%, diferente de usando seleção que não atingiu 29%. Sobre o aumento de sentenças, deduz-se que não ajudou, já que com 366 sentenças, o NN_1 havia atingido 40%.

Por outro lado, o NB teve desempenho melhor com o aumento de número de sentenças. Na tabela 10, observa-se que o maior valor atingido pelo NB foi de 34%, além da maioria dos valores estarem na casa dos 10%, 20%. Agora, o maior valor atingido foi 41% e enxerga-se poucos valores menores que 30%. Todavia, da mesma forma, o NB mostra pior desempenho com n-gramas diferente de 1 e, a redução excessiva do valor de k, 250, por exemplo, apresentam os piores resultados, o 25% do NB_2. Usar a seleção de atributos é eficiente com o k superior a metade da quantidade de *features*. O NB_1, teve melhor performance que o NB_2, já que foi ele quem atingiu a marca de 41% e não mostrou valores abaixo de 30%.

O aumento de sentenças também prejudicou o desempenho do DT. Quando eram 366 sentenças, o DT teve *f1-score* de até 55%. No corpus atual, não passa de 50%. As maiores pontuações do DT_2 aparecem com n-grama igual a 3. O DT_1 parece indiferente quanto ao valor de n-grama. Quanto à variação de quantidade de *features* selecionadas, os melhores valores do DT estão para k menor ou igual a 1000.

Quanto à comparação de qual algoritmo teve melhor performance, ela se mantém quase igual ao corpus anterior. A única alteração é que o NB passa à frente de NN. Portanto, no geral, para o corpus *466*, o que se saiu melhor foi o SVM, seguido do DT, depois o NB e, por último, o NN.

Seleciona-se, novamente, os algoritmos com os parâmetros e atributos que resultaram na melhor e na pior performance. SVM_1 com 57% e NN_2 com 25% (o NB_2 também possui essa pontuação, mas o NN_2 foi escolhido por ter sido o pior algoritmo no geral). Pega-se então os valores de *precision*, *recall* e *f1-score* desses dois algoritmos para cada *label* (tabela 14) e as matrizes de confusão de cada algoritmo (tabela 15).

Das 128 execuções, todas foram de valor zero para a *label O*. Na tabela 15, observa-se que nenhum dos dois algoritmos classificou sentença alguma como sendo de estrutura. Isso pode ser devido à pequena quantidade de sentenças *O*, que é de apenas uma. A segunda classe com menor quantidade de sentenças é a *C*, com 20, outra que nem SVM_1, nem NN_2 conseguiram classificar, com *f1-score* igual a 0,00.

	SVM_1 (ngram=1, k=500)			NN_2 (ngram=3, k=500)		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
B	0,63	0,96	0,76	0,42	0,88	0,57
C	0,00	0,00	0,00	0,00	0,00	0,00
G	0,50	0,03	0,05	0,00	0,00	0,00
M	0,58	0,32	0,41	0,08	0,02	0,03
O	0,00	0,00	0,00	0,00	0,00	0,00
P	0,83	0,51	0,64	0,00	0,00	0,00
R	0,57	0,63	0,60	0,19	0,13	0,15
avg/total	**0,60**	**0,62**	**0,57**	**0,21**	**0,37**	**0,25**

Tabela 14 – Relatório de precision, recall *e* f1-score *para cada* label

Outra performance 0,00 é do NN_2 para *labels G* (lacuna), que são quase 8% das sentenças e SVM_1 alcançou apenas a pontuação de 5%. Uma última performance 0,00, é do NN para *P* (propósito), em que o SVM foi a segunda maior pontuação, 64%. As três maiores pontuações do SVM são para *B* (contexto), *P* (propósito) e *R* (resultados), que são as sentenças com maiores números de sentenças. Essas três classes juntas compõem 75,1% do corpus. Todavia, a quantidade de sentenças não é diretamente proporcional à performance, por exemplo, o *P*, que desses três é o que tem menos, as performances dos algoritmos ficaram em segundo, e do *R*, que em quantidade é o segundo, ficou em terceiro, como se observa na tabela 14. Para o NN, as duas melhores performances são para as duas *labels* com maiores quantidades de sentenças, contexto e resultados. O NN obteve 57% e 15%, respectivamente.

A tabela 15 permite a observação de que a maior tendência de classificação ficou para sentenças de contexto, ao invés de resultados como aconteceu para o corpus *366*.

SVM_1 (ngram=1, k=500)								NN_2 (ngram=3, k=500)						
171	0	1	1	0	0	6	*B*	158	0	0	2	0	2	17
5	0	0	1	0	1	13	*C*	11	0	0	1	0	0	8
33	0	1	0	0	0	2	*G*	33	0	0	0	0	0	3
18	1	0	19	0	2	19	*M*	44	0	0	1	0	0	14
1	0	0	0	0	0	0	*O*	1	0	0	0	0	0	0
21	0	0	3	0	35	9	*P*	54	0	0	2	0	0	12
24	1	0	9	0	4	65	*R*	79	2	0	6	0	3	13

Tabela 15 – Matriz de confusão de SVM_1 (ngram=1, k=500) e NN_2 (ngram=3, k=500)

4.5. Resultados do corpus *832*

A tabela 16 mostra as médias dos *f1-scores* dos algoritmos SVM, NN, NB e DT. Nela, observa-se que a maior pontuação ainda é do SVM_1 e com 57%, assim como nos corpora *366* e *466*. De fato, o aumento de 466 para 832 fez com que o desempenho do SVM fosse melhor, mas não mostrou uma melhora significativa.

Sob o aumento de sentenças, o NN mostrou um alcance. O NN_2 foi capaz de atingir 40%, como se vê na tabela 16. Sem seleção de atributos, o NN_2 mostrou melhores performances que o NN_1. Já com seleção de atributos, ambos mostraram performances equivalentes, exceto para n-grama igual a 3 e *k* baixo (500), em que o NN_1 alcançou 38% e o NN_2 apenas 30%.

ngram	k	SVM_1	SVM_2	NN_1	NN_2	NB_1	NB_2	DT_1	DT_2
1	sem	0,47	0,47	0,34	0,40	0,33	0,32	0,43	0,41
	2500	0,52	0,52	0,28	0,27	0,36	0,34	0,42	0,40
	1000	0,53	0,54	0,26	0,24	0,38	0,37	0,41	0,42
	500	0,57	0,55	0,28	0,26	0,37	0,35	0,48	0,47
	250	0,56	0,53	0,28	0,28	0,36	0,35	0,49	0,48
2	sem	0,46	0,47	0,34	0,37	0,31	0,28	0,43	0,40
	10000	0,47	0,47	0,25	0,25	0,33	0,30	0,43	0,41
	5000	0,51	0,49	0,24	0,25	0,32	0,29	0,42	0,42
	2500	0,54	0,51	0,27	0,26	0,30	0,29	0,48	0,46
	1000	0,53	0,49	0,26	0,28	0,29	0,29	0,47	0,48
	500	0,50	0,47	0,28	0,31	0,27	0,28	0,47	0,48
3	sem	0,42	0,43	0,34	0,37	0,27	0,26	0,41	0,41
	25000	0,42	0,43	0,22	0,26	0,30	0,27	0,42	0,40
	10000	0,46	0,45	0,35	0,24	0,38	0,25	0,40	0,42
	5000	0,51	0,47	0,37	0,26	0,36	0,25	0,48	0,46
	2500	0,50	0,46	0,36	0,28	0,35	0,25	0,46	0,48
	1000	0,46	0,43	0,37	0,30	0,34	0,24	0,47	0,48
	500	0,41	0,41	0,38	0,30	0,34	0,25	0,46	0,45

Tabela 16 – Média de f1-score *dos algoritmos para o corpus* 832

Já o NB teve uma queda de performance ao aumentar de 466 para 832 sentenças. Ainda mantém resultados acima de 30% em sua maioria, mas não alcançou os 41% do NB_1 (tabela 13). Como vem ocorrendo, as melhores performances do NB estão para unigramas.

Novamente, a performance do DT caiu ao aumentar o número de sentenças. Manteve pontuações acima de 0,40, porém não alcançou nem 0,55 como no corpus *366*, nem 0,50 como no *466*. Manteve-se indiferente à variação de n-grama e melhor com seleção de *features* para *k* menor ou igual à 1/3 do total de *features* para determinado n-grama.

No geral, a ordem se mantém como no *466*. O SVM se mostrou como sendo o melhor algoritmo, seguido do DT. Em terceiro, o NB e, por último o NN.

Nas tabelas 17 e 18, é mostrado o relatório de *precision*, *recall* e *f1-score* para cada classe e as matrizes de confusão, respectivamente. Foram selecionados o SVM_1 (unigrama e k=500) e NN_1 (trigrama, k=2500), a maior e pior performance.

	SVM_1 (ngram=1, k=500)			**NN_1 (ngram=3, k=25000)**		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
B	0,63	0,89	0,74	0,34	0,90	0,49
C	0,50	0,03	0,05	0,00	0,00	0,00
G	0,80	0,11	0,20	0,00	0,00	0,00
M	0,58	0,25	0,35	0,00	0,00	0,00
O	0,00	0,00	0,00	0,00	0,00	0,00
P	0,73	0,73	0,73	0,00	0,00	0,00
R	0,55	0,70	0,62	0,35	0,22	0,27
avg/total	**0,62**	**0,62**	**0,57**	**0,32**	**0,39**	**0,22**

Tabela 17 – Relatório de precision, recall *e* f1-score *para cada* label

Assim como nos dois corpora anteriores, nenhum dos dois algoritmos classificaram sentença alguma como estrutura. Das 144 execuções dos algoritmos, apenas em 10, obteve-se valores diferentes de zero. Nove deles foram do DT_2 e variou de 0,13 a 0,46.

O NN_1 pontuou apenas em contexto e resultados, 49% e 27% (tabela 17). Essas duas classes são as com maior número de sentenças, somando as duas temos 57,21% das sentenças. Na tabela 18, observa-se que houveram apenas 21 tentativas de predição que não foram em *B* ou *R*, e, dessas 21 tentativas, 15 foram em *C* e 6 em *M*. Ou seja, o NN não classificou sentença alguma como sendo *G*, *O* ou *P*.

Já o SVM, obteve zeros apenas em estrutura. Nas sentenças de conclusão, que é a segunda em quantidade de sentenças, o *Support Vector Machine* obteve um *f1-score* de apenas 5%. Na tabela 18, vê-se que o SVM classificou 352 sentenças como sendo *B*, 280 como *R* e 133 como *P*. Assim como no corpus anterior, as maiores pontuações estão em contexto, propósito e resultados, nesta ordem. Quanto à quantidade de sentenças, a ordem é contexto, resultados e propósito (tabela 2).

SVM_1 (ngram=1, k=500)								**NN_1 (ngram=3, k=25000)**						
228	0	1	1	0	8	18	*B*	230	2	0	1	0	0	23
11	1	0	0	0	1	27	*C*	21	0	0	0	0	0	19
54	1	8	0	0	1	8	*G*	68	0	0	0	0	0	4
18	0	0	26	0	7	53	*M*	80	0	0	0	0	0	24
0	0	0	0	0	2	5	*O*	5	0	0	0	0	0	2
14	0	0	7	0	97	15	*P*	111	4	0	2	0	0	16
37	0	1	11	0	17	154	*R*	160	10	0	2	0	0	48

Tabela 18 – Matriz de confusão de SVM_1 (ngram=1, k=500) e NN_1 (ngram=3, k=25000)

4.6. Resultados do treino com *466* e teste em *366*

Na tabela 19, são apresentadas as médias dos *f1-scores* dos algoritmos SVM, NN, NB e DT. Neste teste, os algoritmos foram treinados com o corpus *466* e testados no *366*. Pode-se dizer que é outro teste em cima do *832*, porém sem *cross-validation*, por isso, será feito uma comparação maior com os resultados presentes no tópico anterior (4.5).

Para o SVM, esta estratégia teve resultados piores do que utilizando a validação cruzada. Antes, eram alcançadas performances de até 57%, agora, o máximo é 44%. Além disso, antes, não haviam performances abaixo de 40%, agora, observa-se resultados de 30%. Algo que se manteve foi que as melhores performances se apresentam ao ser utilizado a seleção de atributos na média de 14% das *features*.

Já para o NN, o uso de seleção de *features* prejudicou sua performance, assim como se observa na tabela 16. Os maiores valores são 35%, 36% para NN_1 e 23%, 22% para NN_2, sem seleção de atributos. Ao utilizar a seleção, as performances caem para 9%, 8% (menor valor da tabela 19). Por outro lado, a variação de n-gramas não mostrou grandes influências no resultado, variação de 1%.

ngram	K	SVM_1	SVM_2	NN_1	NN_2	NB_1	NB_2	DT_1	DT_2
1	sem	0,43	0,39	0,35	0,23	0,25	0,27	0,36	0,35
	1000	0,40	0,37	0,12	0,10	0,26	0,25	0,39	0,35
	500	0,44	0,36	0,11	0,09	0,29	0,22	0,42	0,36
	250	0,43	0,37	0,10	0,09	0,27	0,15	0,45	0,34
	100	0,40	0,33	0,12	0,12	0,20	0,12	0,44	0,36
2	sem	0,39	0,40	0,36	0,22	0,22	0,24	0,35	0,36
	5000	0,35	0,35	0,10	0,11	0,20	0,23	0,35	0,31
	2500	0,41	0,38	0,12	0,09	0,28	0,25	0,43	0,39
	1000	0,42	0,39	0,09	0,09	0,28	0,25	0,39	0,37
	500	0,43	0,34	0,11	0,09	0,24	0,18	0,43	0,39
3	sem	0,36	0,36	0,36	0,22	0,21	0,24	0,34	0,34
	10000	0,31	0,33	0,08	0,11	0,19	0,23	0,36	0,28
	5000	0,39	0,37	0,08	0,12	0,27	0,24	0,41	0,37
	2500	0,43	0,38	0,09	0,08	0,28	0,24	0,44	0,38
	1000	0,41	0,35	0,12	0,12	0,25	0,21	0,45	0,38
	500	0,30	0,29	0,21	0,14	0,21	0,14	0,39	0,29

Tabela 19 – Média de f1-score *dos algoritmos treinados com* 466 *e testados em* 366

Para o NB, também se tem uma queda de performance, comparando com o teste anterior (com o corpus *832*). A variação de n-gramas não favorece, nem prejudica, na performance do NB, assim como aconteceu usando *cross-validation*. Já a seleção de *features*, depende do valor *k* que não deve ultrapassar de 14% do total de atributos.

Por fim, o DT também teve uma queda de resultados, porém não tanto quanto os outros algoritmos. É o algoritmo que alcançou o melhor resultado, 45% com o DT_1. Assim como no caso do *832* com validação cruzada, o DT se sai melhor ao usar seleção de *features* com *k* não ultrapassando de 9% do total de *features*.

Da mesma forma que vem acontecendo, no geral, os algoritmos usando maior quantidade de atributos tiveram performances melhores do que utilizando apenas *tf-idf* da palavra e posição. Utilizar a validação cruzada mostrou ser uma estratégia melhor do que não usá-la. Sem usar *cross-validation*, o DT (DT_1) teve a melhor performance ao atingir 0,45. Em segundo, com 0,44 vem o SVM. Depois, vem o NB e, por último, o NN com a pior performance de 0,08. Selecionando a melhor e a pior performance, compõe-se os dados das tabelas 20 e 21.

	DT_1 (ngram=3, k=1000)			NN_2 (ngram=3, k=2500)		
	precision	*recall*	*f1-score*	*precision*	*recall*	*f1-score*
B	0,41	0,70	0,51	0,21	1,00	0,35
C	0,20	0,05	0,08	0,00	0,00	0,00
G	0,54	0,19	0,29	0,00	0,00	0,00
M	0,30	0,29	0,29	0,00	0,00	0,00
O	0,00	0,00	0,00	0,00	0,00	0,00
P	0,73	0,66	0,69	0,00	0,00	0,00
R	0,48	0,45	0,46	1,00	0,02	0,03
avg/total	**0,47**	**0,47**	**0,45**	**0,36**	**0,22**	**0,08**

Tabela 20 – Relatório de precision, recall *e* f1-score *para cada* label

Na tabela 20, estão as performances dos algoritmos DT_1 e NN_2 para cada *label*. Da mesma maneira, as performances para estrutura (*label O*), de ambos os algoritmos, são 0,00. O DT_1 classificou apenas uma sentença como estrutura e o NN_2 não classificou sentença alguma como *O*. Isso se deve ao fato da pequena quantidade de sentenças *O* (essa classe tem apenas uma sentença para treino e seis para classificar).

Na tabela 21, estão as matrizes de confusão de DT_1 e NN_2. Observa-se que o DT_1 classificou a maioria, 36,33%, das sentenças como *B* e 30,32% como *R*. Porém, na tabela 20, é mostrado que o DT_1 teve maior pontuação com sentenças de propósitos que, para treino, são 68 sentenças. O algoritmo classificou 16,12% das sentenças como *P* e obteve performance de 69%.

DT_1 (ngram=3, k=1000)								NN_2 (ngram=3, k=2500)						
54	0	3	4	0	2	14	*B*	77	0	0	0	0	0	0
9	1	0	2	0	0	8	*C*	20	0	0	0	0	0	0
17	0	7	6	0	0	6	*G*	36	0	0	0	0	0	0
5	1	1	13	1	1	23	*M*	45	0	0	0	0	0	0
3	0	1	0	0	2	0	*O*	6	0	0	0	0	0	0
10	0	1	4	0	43	7	*P*	65	0	0	0	0	0	0
35	3	0	15	0	11	53	*R*	115	0	0	0	0	0	2

Tabela 21 – Matriz de confusão de DT_1 (ngram=3, k=1000) e NN_2 (ngram=3, k=2500)

Além dos zeros nas sentenças de estrutura, o NN_2 teve zeros nas sentenças de conclusão, lacuna, metodologia e propósito, como se observa na tabela 20. Isso, porque, como é visto na tabela 21, o NN_2 não classificou sentença alguma como *C*, *G*, *M*, *O* ou *P*. Como *label* de resultados, o NN_2 classificou apenas duas de forma correta e o restante das 364 sentenças foram classificadas como contexto.

5. Conclusões

Sobre os corpora em inglês dos resumos da PubMed, não foi possível executar os classificadores que utilizavam os algoritmos *Nearest Neighbors* e *Decision Trees*, devido à limitação de memória da máquina utilizada. Pelo mesmo motivo, para o corpus *data* não foi possível executar para trigramas.

Utilizar mais atributos como *tf-idf* da palavra, *tf-idf* da palavra anterior, *tf-idf* da palavra posterior e a posição da palavra mostrou melhores resultados do que utilizar apenas o *tf-idf* da palavra e posição da palavra. Apenas o algoritmo NN mostrou-se melhor sem a utilização de seleção de *features*. Os outros três algoritmos atingiram melhores pontuações utilizando a seleção pela distribuição *chi-squared* que não ultrapassasse 15% das características totais para determinado n-grama. Com relação à variação de n-gramas, a maioria dos melhores resultados estão para unigramas e bigramas. Quanto à variação de sentenças, para os corpora em inglês, o aumento de sentenças melhorou, mesmo que minimamente, a performance dos algoritmos. Já para os corpora em português, o aumento de sentenças fez com que a performance caísse. Por fim, o uso do estimador de validação cruzada, observada apenas no corpus *832*, apresentou melhores resultados do que sem o seu uso.

No geral, o algoritmo *Support Vector Machine* se mostrou melhor, na maioria das vezes alcançou os melhores resultados dentre os quatro algoritmos. Individualmente para cada *label*, um algoritmo é melhor para sua classificação, por exemplo, nos corpora em português, para sentenças de contexto e resultado, o melhor algoritmo foi o SVM; já para conclusão, lacuna e estrutura foi o DT (o restante das classes ou era SVM ou era DT).

Algo que não é possível ser afirmado é se a melhor performance para os corpora *dev* e *data* é dado apenas pelo maior número de sentenças ou se o idioma dos corpora também influenciou.

6. Bibliografia

LORENA, A. C.; CARVALHO, A. C. P. L. F. de. Introdução às Máquinas de Vetores Suporte (*Support Vector Machine*). Relatórios Técnicos do ICMC, Nº 192, São Carlos, 2003.

REZENDE, S. O. Sistemas Inteligentes: Fundamentos e Aplicações. Editora Manole Ltda, 1ª Edição, 2003.

ROMEIRO, A. K. Q. Um estudo sobre o uso da teoria da estrutura retórica (RST) para sumarizar a sabedoria da coletividade. 2016. 109 f.. Dissertação (Mestrado em Computação) - Universidade Federal Fluminense, Niterói – RJ, 2016.

WEBBER, B.; EGG, M.; KORDONI, V. Discourse structure and language technology. Natural Language Engineering, Volume 18, Issue 04, 2011, pp. 437-490.

WEISSBERG, R.; BUKER, S. Writing up research: experimental research report writing for students of english. Englewood Clis, NJ: Prentice Hall Regents, 1990.

7. Anexos (referentes à pesquisa): comprovantes de apresentação dos resultados da pesquisa em eventos científicos, e/ou comprovante de publicação, com a participação do (s) acadêmico (s) em periódicos indexados e/ou com corpo editorial (conforme o regulamento do Programa PIC)