Classificação não-supervisionada hierárquica de artigos jornalísticos

Cirillo Ribeiro Ferreira
Orientador: Prof. Dr. Alair Pereira do Lago

9 de fevereiro de 2015

Sumário

1 Introdução

A tarefa de classificar e agrupar documentos textuais remonta desde a antiguidade, iniciando-se na criação da primeira biblioteca do mundo em Nínive, Assíria (atual Iraque), por volta do século VII a.C [1]. Desde então, profissões como bibliotecário, documentalista e arquivista foram criadas para atuar na organização desses documentos que vem sendo produzidos pela humanidade.

Porém com a criação da internet e a popularização de seu uso como ferramenta de comunicação, houve uma explosão de informação que tornou praticamente impossível a classificação desses novos tipos de documentos de maneira manual.

A área de classificação de documentos é de grande interesse e possui diversas aplicações práticas como classificação de *spam*, identificação do idioma utilizado e análise de sentimento. Em especial, a classificação de artigos jornalísticos oferece um enorme desafio devido à grande quantidade de novos documentos criados diariamente e a diversidade de temas abordados, especialmente em blogs, mas que carecem de melhor organização.

1.1 Motivação

A motivação desse trabalho vem da insatisfação com a falta de tempo para uma leitura mais crítica e ciente de um contexto maior devido à enorme quantidade de notícias que estamos sujeitos a receber todos os dias e da organização simplória feita pelos grandes portais de notícias.

O motivo de se utilizar algoritmos não-supervisionados vem da necessidade de encontrar relações implícitas entre os artigos e também da diversidade e quantidade de artigos disponíveis para classificação, uma vez que fica quase que impossível criar manualmente um conjunto de treinamento para subsídio de um algoritmo supervisionado.

Ao contrário da classificação supervisionada, a classificação não-supervisionada não necessita de um conjunto de treinamento como entrada, permitindo o seu uso em conjuntos de dados bem variados, algo que é bem comum em artigos jornalísticos.

A ideia não é separar os artigos jornalísticos em grupos muito generalistas comumente usados em jornais ou portais de notícias, como por exemplo as seções: Brasil, Mundo, Economia e Esportes; mas encontrar grupos mais intrínsecos que representem com mais acurácia a informação passada por esses artigos.

Segundo Martin Ester *et al.* [7], existem alguns desafios da área de agrupamento de documentos como: alta dimensionalidade da coleção, onde cada palavra distinta na coleção constitui uma dimensão e quanto maior a dimen-

são, maior o desafio para a construção de um algoritmo escalável e eficiente; grande volume de dados; busca de alta precisão e consistência, pois dependendo do tipo de documentos, alguns algoritmos apresentam resultados muito abaixo do esperado; e descrição mais significativa dos grupos, pois facilita a utilização pelos usuários.

1.2 Objetivos

O trabalho de classificação não-supervisionada dos artigos jornalísticos foi dividido em duas partes:

- Criação de uma biblioteca para agrupamento de artigos jornalísticos.
- Proposta e implementação do sistema hVINA (*Hierarchical Viewer of News Articles*) para análise de agrupamento de artigos jornalísticos.

A representação hierárquica de um agrupamento de documentos é geralmente feita através de um dendrograma, como na figura 1.

Figura 1: Esquema do resultado esperado

1.3 Organização do trabalho

Na seção 2 é apresentado os fundamentos da área de reconhecimento de padrão e mais especificamente de análise de agrupamento. Na seção 3 é

discutido em detalhe o algoritmo FIHC, escolhido para ser implementado na biblioteca e sistema hVINA propostos na seção 4 para o agrupamento de artigos jornalísticos. Na seção 5 são apresentados os resultados obtidos com essas ferramentas. Na seção 6 são apresentadas as conclusões finais do trabalho. E por fim, na seção 7 é apresentada uma análise subjetiva do curso do Bacharelado em Ciência da Computação.

2 Fundamentos

Antes de apresentar e discutir sobre a biblioteca desenvolvida e o sistema proposto, serão apresentados alguns conceitos básicos necessários para a compreensão do conteúdo e dos resultados obtidos.

2.1 Reconhecimento de padrão

Segundo Theodoridis e Koutroumbas [2], reconhecimento de padrão é uma área da ciência cujo objetivo é a classificação de objetos dentro de um número de categorias ou classes. Esses objetos de estudo variam de acordo com cada aplicação, podem ser imagens, sinais em forma de ondas (como voz, luz, rádio) ou qualquer tipo de medida que necessite ser classificada. As implicações práticas desses estudos permeiam todas as áreas de atuação humana, principalmente numa sociedade pós-industrial, onde a necessidade de automação das diversas atividades é essencial. Sua aplicação vai desde sistemas que utilizam visão computacional, passando por reconhecimento de caracteres até sistemas para auxílio à tomada de decisão.

2.1.1 Formas de processo de aprendizagem

Na área de reconhecimento de padrão os algoritmos utilizam técnicas de aprendizagem computacional para a realização de alguma tarefa e eles são geralmente classificados de acordo a necessidade de intervenção humana durante alguma fase de aprendizagem do algoritmo. São elas: aprendizagem supervisionada, aprendizagem não-supervisionada e aprendizagem semi-supervisionada.

A classificação supervisionada consiste na utilização de um conjunto de dados previamente classificados (ou anotados), que servirá na construção de um modelo para a classificação de novos objetos. Ou seja, algoritmos que utilizam esse conjunto de dados fazem uso de um conhecimento a priori, originado de uma fonte externa. Esse conjunto de dados inicial é denominado de conjunto de treinamento (*training set*). Porém, em muitos casos a criação de um conjunto de treinamento é inviável devido à natureza do problema. Para esses tipos, é então empregado a classificação não-supervisionada, que não faz uso de um conjunto de treinamento, pois o próprio algoritmo busca descobrir similaridades intrínsecas nos objetos e agrupa aqueles objetos mais similares. [2, 3, 4]

Por fim, a classificação semi-supervisiona emprega técnicas da classificação supervisionada e não-supervisionada, geralmente com um conjunto pequeno de treinamento.

No contexto de reconhecimento de padrão, é comum chamar a classificação supervisionada de apenas classificação e a classificação não-supervisionada de análise de agrupamento, ou simplesmente agrupamento (do inglês *clustering*).

2.2 Análise de agrupamento

Análise de agrupamento é uma classificação de padrão que emprega o processo de aprendizagem não-supervisionada e tem como objetivo o particionamento de objetos em grupos cujos membros sejam similares entre si e diferentes dos membros de outros grupos [3]. Porém, também tem os objetivos secundários de evitar grupos muito pequenos ou muito grandes e encontrar grupos que façam sentido ao usuário. Segundo Theodoridis e Koutroumbas [2], a análise de agrupamento é largamente utilizada na resolução de diversos problemas, e pode ser encontrada em outras áreas por nomenclaturas diferentes, como taxonomia numérica em biologia, e tipologia na área das ciências sociais.

Um grande problema enfrentado na análise de agrupamento é a dimensionalidade ou variáveis dos dados envolvidos, sejam eles observações de estrelas no céu, observações de organismos vivos na natureza ou documentos textuais disponíveis na internet, e uma técnica bastante utiliza para tentar solucionar tal problema é a redução dimensional através da extração ou seleção de características, que envolve em sintetizar os dados em características mais relevantes para o problema principal, tentando minimizar a perda de informação.

2.3 Agrupamento de documentos textuais

Uma aplicação da análise de agrupamento é na organização de documentos. Geralmente ao fazer alguma consulta em sistemas de busca são retornadas centenas de documentos que possuem algum tipo de relevância à consulta, porém esse grande número de resultado pode inviabilizar a utilização desses sistemas, por isso a maioria deles utiliza algumas técnicas de análise de agrupamento para organizar automaticamente os resultados em categorias que fazem sentido ao usuário.

Em especial, quando se fala em documentos textuais, o processo de redução dimensional através de extração de características dos dados é de sensível importância devido à grande redundância dos dados, que neste caso, são sentenças ou palavras. Processar os documentos sem antes usar uma boa técnica de redução dimensional acarreta quase sempre em baixa eficiência da solução.

2.4 Tipos de algoritmo de agrupamento

Um algoritmo de agrupamento objetiva encontrar as classes naturais das observações, baseados em alguma similaridade. Existem diversos algoritmos para agrupamento desenvolvidos [2, 3], porém neste capítulo serão citadas duas categorias de algoritmos mais empregados nesse processo. Uma explicação com aspectos mais detalhados desses métodos pode ser encontrada em [2, 3].

2.4.1 Agrupamento plano

Agrupamento plano é a categoria de algoritmos cujo os grupos resultantes são dados num único nível, onde nenhum grupo tem relação com outro. O principal algoritmo dessa categoria é o *k-means* [4].

Antes de falar do *k-means* é preciso relembrar um ponto essencial ao entendimento desse algoritmo, que é a definição de centroide. Na geometria, o centroide é o ponto central de uma figura. Dessa forma, dado um grupo, se ligarmos com um segmento de linha as observações mais distantes podemos visualizar uma figura geométrica, e o centroide desse grupo seria a observação mais próxima ao centro dessa figura.

Em termos gerais, o objetivo do *k-means* é dividir n observações em k grupos, minimizando a média das distâncias euclidianas quadráticas entre as observações e os respectivos centroides dos grupos, ou seja, cada observação é associada ao grupo cujo o centroide está mais próximo.

O algoritmo inicia com uma escolha aleatória dos k centroides e a cada iteração é feito um refinamento na escolha desses centroides até que não haja mais variação na escolha dos centroides ou até que se atinja um número predefinido de iterações. A figura 2 exemplifica a sequência descrita acima.

Há duas propriedades importantes do *k-means*:

- Cada objeto deve pertencer ao menos um dos k grupos.
- Nenhum objeto pertence a mais que um grupo.

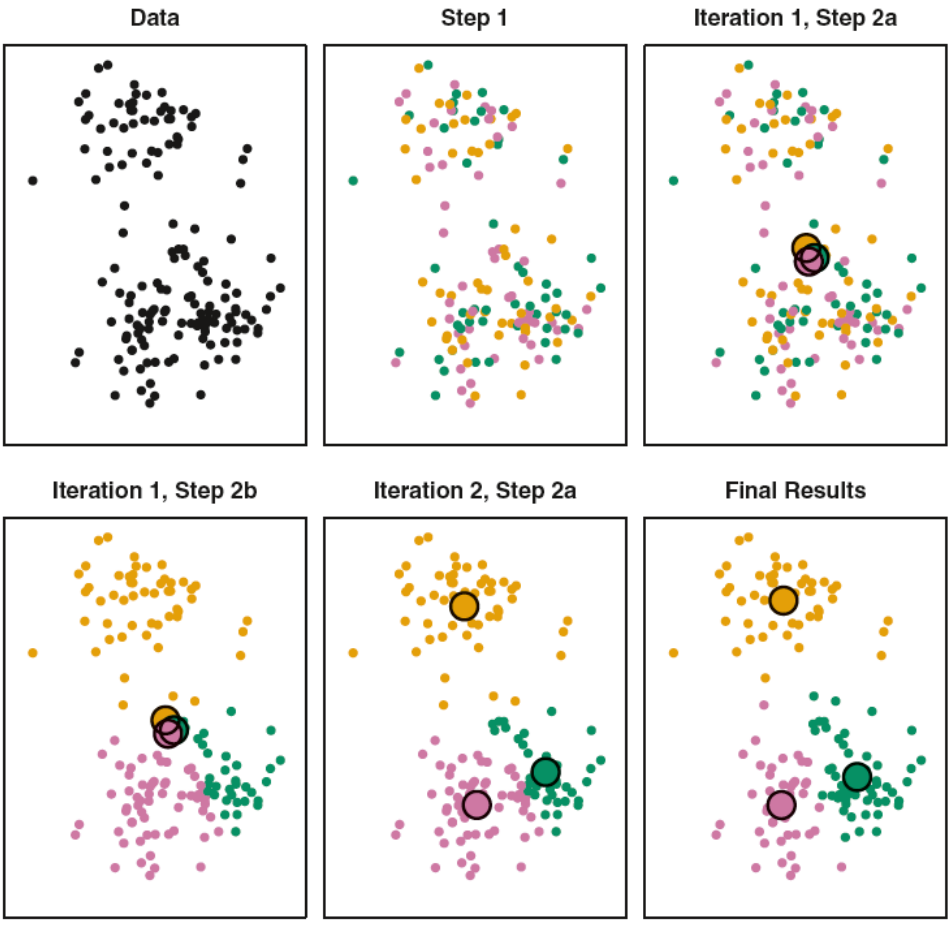

Figura 2: Sequência de passos do algoritmo k-means, retirado de [14]

Um cuidado especial na utilização desse algoritmo é na escolha do número de grupos desejado. Uma escolha inadequada poderá resultar em um mau agrupamento, unindo duas classes naturais em um único grupo ou dividindo uma classe natural em dois grupos, porém algumas técnicas tem sido desenvolvidas para resolver esse problema [3, 4]. Um exemplo de problema que pode ocorrer está ilustrado pela figura 3.

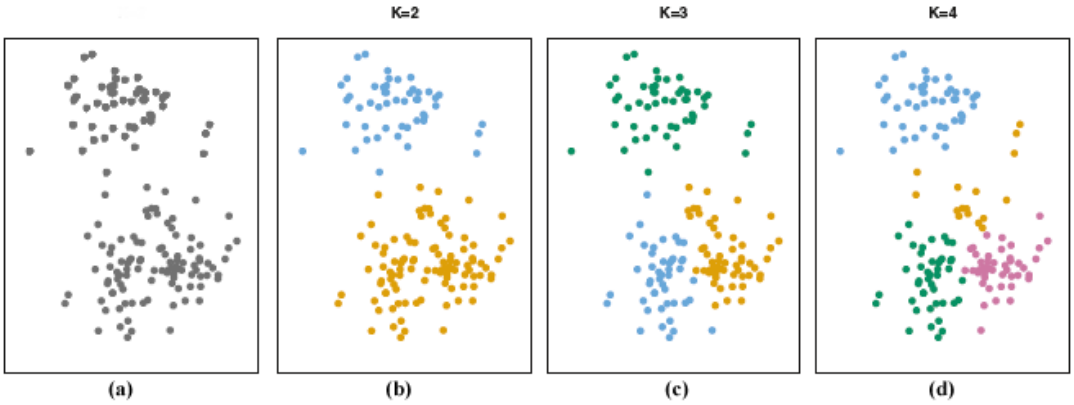

Figura 3: Em (a) é mostrado um conjunto de dados fictício com três classes naturais. Em (b) quanto é escolhido o número de grupos k = 2, as classes naturais 2 e 3 são unidas num único grupo. Em (c), com k = 3, os grupos refletem as classes naturais dos dados, e em (d) com k = 4 a classe natural 3 é dividida entre dois grupos.

Conforme John Wang [9], o algoritmo *k-means* não é adequado para descobrir grupos de tamanhos que variam muito, um cenário comum em agrupamento de documentos. Além disso, é sensível a ruídos que pode ter uma grande influência sobre a escolha do centroide de um grupo, que por sua vez diminui a precisão do agrupamento. Todavia algoritmos como o *k-medoids* [9] foram propostos para resolver o problema do ruído.

2.4.2 Agrupamento hierárquico

Agrupamento plano geralmente é mais simples e fácil de implementar, porém existem certos tipos de problemas em que a visualização de um agrupamento de um único plano não é suficientemente útil, além disso, como visto no tópico anterior, esse tipo de agrupamento tem dois problemas que são a escolha do número de grupo na entrada e o fato de não serem determinísticos [4]. Dessa forma, algoritmos que criassem agrupamentos com vários níveis e que tivessem como requisito opcional a entrada do número de grupos foram desenvolvidos. Esses algoritmos criam uma árvore hierárquica de grupos, uma estrutura que gera mais informação, uma vez que as relações implícitas entre os grupos ficam mais evidentes.

Existem duas abordagens no funcionamento de um algoritmo hierárquico que são:

- Abordagem aglomerativa:. Gera a árvore de grupo ao continuamente

calcular a similaridade de todos os pares de grupos e unificar o par mais similar (segundo um critério de similaridade), criando a árvore das folhas até a raiz, por isso é também conhecida como abordagem bottom-up. Um exemplo dessa abordagem é ilustrado na figura 4.

- Abordagem divisiva: Gera a árvore de grupo da raiz às folhas (top-down), utilizando em cada nível da árvore um algoritmo de agrupamento plano para a divisão dos grupos em nós filhos.

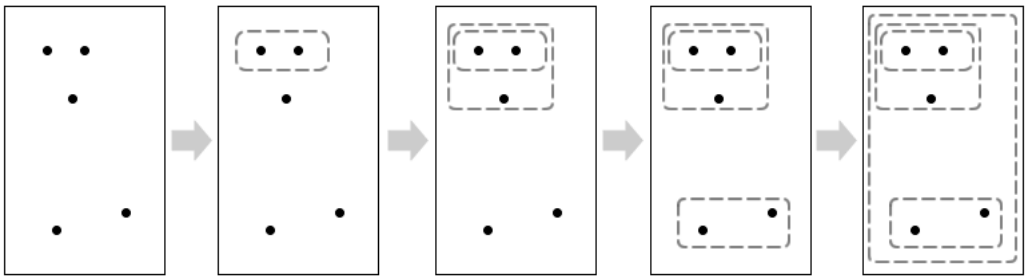

Figura 4: Exemplo de um algoritmo hierárquico aglomerativo, retirado de [8], onde o critério de similaridade utilizado é a distância euclidiana.

2.5 Critério de avaliação

Existem 2 tipos de erros que podem ocorrer no processo de agrupamento. O primeiro, chamado de falso positivo (FP) ou erro tipo I, ocorre quando o algoritmo associa dois documentos não similares a um mesmo grupo, e o segundo, chamado de falso negativo (FN) ou erro tipo II, ocorre quando o algoritmo associa dois documentos similares a grupos distintos. Um critério de avaliação justo tem de levar em conta as ocorrências desses dois erros. A tabela 1 a seguir, mostra a matriz de contingência entre os documentos e grupos.

	Mesmo grupo	Diferentes grupos
Documentos similares	Verdadeiro positivo (VP)	Falso negativo (FN) ou Erro tipo II
Documentos não similares	Falso positivo (FP) ou Erro tipo I	Verdadeiro negativo (VN)

Tabela 1: Matriz de contingência

Duas medidas importantes na área de reconhecimento de padrão e também na estatística, são a precisão e cobertura do resultado observado. A precisão mede a fração de documentos que estão classificados corretamente no grupo e é dado pela seguinte fórmula:

$$P = \frac{VP}{VP + FP} \tag{1}$$

Já a cobertura mede a fração de documentos que foram classificados corretamente dentre os documentos similares e é dado pela fórmula:

$$R = \frac{VP}{VP + FN} \tag{2}$$

Por exemplo, seja dado o conjunto de documentos {1, 2, 3, 4, 5, 6}, onde N_1 = {1, 3, 5} fazem parte de uma classe natural, ou seja, são similares; e N_2 = {2, 4, 6} fazem parte de outra classe natural. Dado um grupo C = {1, 3, 4, 5} a precisão de C em relação a N_1 é P = 3/4 e a cobertura é R = 3/3 = 1, ou seja, nem todos os documentos de C são similares entre si, apesar de que todos os documentos de N_1 foram agrupados corretamente em C.

Porém, a utilização de duas medidas para avaliação e comparação de algoritmos torna difícil determinar se um algoritmo é superior a outro ou não. Pois é melhor um algoritmo com alta precisão e baixa cobertura, ou vice-versa? A resposta para essa pergunta depende do contexto onde é aplicado o agrupamento. Segundo Manning et al, na maioria das vezes é mais aceitável o erro de tipo I, pois separar documentos similares em grupos distintos é geralmente pior do que juntar documentos não similares num mesmo grupo [7].

Dado isto, um critério para avaliação bastante utilizado é a medida F (*F-measure*), pois avalia o agrupamento utilizando a média harmônica ponderada da precisão e da cobertura. A medida F é dada por:

$$F_\beta = (1 + \beta^2) \times \frac{P \times R}{\beta^2 P + R} \tag{3}$$

onde $\beta^2 \in [0, \infty]$.

Um valor para β bastante utilizado é $\beta = 1$, pois tanto a precisão quanto a cobertura terão o mesmo peso na medida. Neste caso, a medida seria simplificada da seguinte forma:

$$F_1 = 2 \times \frac{P \times R}{P + R} \tag{4}$$

Já definindo $\beta > 1$, a precisão será mais enfatizada em relação à cobertura. Note que para a medida F, o algoritmo não basta ter apenas boa

precisão ou apenas boa cobertura, mas sim ambas características, o que a torna um dos critérios mais utilizados para a análise. Porém, ela não é o único critério de avaliação existente. Dentro da área de estudo sobre avaliação de agrupamentos existem critérios internos e externos e a medida F se enquadra na segunda categoria, pois avalia o agrupamento a partir de uma classificação de referência [7].

3 FIHC

Frequent Itemset-based Hierarchical Clustering (FIHC) é um algoritmo para agrupamento hierárquico de documentos proposto por Martin Ester *et al.* [7] e que utiliza o conceito de conjuntos de itens frequentes (do inglês *frequent itemset*).

Segundo Martin Ester *et al.*, a ideia utilizada para o critério de agrupamento dos documentos é que existe algum conjunto de itens frequentes para cada grupo (tópico) no conjunto de documentos, e diferentes grupos compartilham poucos conjuntos de itens frequentes.

Segundo os autores, o algoritmo tenta resolver alguns desafios da área como: alta dimensionalidade da coleção, onde cada palavra distinta na coleção constitui uma dimensão, e quanto maior a dimensão, maior o desafio para a construção de um algoritmo escalável e eficiente; grande volume de dados; alta precisão e consistência, dependendo do tipo de documentos, alguns algoritmos apresentam resultados muito abaixo do esperado; e descrição mais significativa dos grupos, pois facilita a utilização pelos usuários.

Um ponto de interesse é que, diferentemente dos algoritmos mais comuns para agrupamento hierárquico, no FIHC a parametrização do número de grupos desejado é opcional, assim o usuário não precisa ter nenhum conhecimento prévio da coleção de documentos.

O algoritmo utiliza a abordagem aglomerativa, descrita na seção 2.4.2, porém ao invés de construir a árvore baseando-se na similaridade entre as observações (documentos) ele a faz baseando-se nos grupos [7]. Possui dois passos principais, que é a construção dos grupos e posteriormente a criação da árvore hierárquica, entretanto antes de apresentá-los é necessário introduzir algumas definições.

3.1 O que é conjunto de itens frequentes

Por definição, conjunto de itens é o conjunto de palavras que ocorrem em conjunto na coleção de documentos. Logo conjunto de itens frequentes é o conjunto de palavras que ocorrem numa quantidade de documentos acima de um limiar de suporte definido. Em outras palavras, suponha que l seja o limiar de suporte e que F seja um conjunto de palavras. O suporte (*support* em inglês) de F é a proporção de documentos no qual F é um subconjunto das palavras desses documentos. Assim, dizemos que F é um conjunto de itens frequentes se o seu suporte é l ou superior. A tabela 2 ilustra um exemplo:

Documento	Palavras (itens)
doc.1	{Bolsa, europeia, opera, em, queda, após, anúncio, de, pacote, grego}
doc.2	{Japão, e, Grécia, ficam, no, 0, a, 0, no terceiro, dia, de, competição}
doc.3	{Descoberta, de, tumba, misteriosa, anima, Grécia, em, meio, à, crise, europeia}
doc.4	{Grécia, assume, presidência, da, União, Europeia, por, seis, meses}
doc.5	{Crise, está, prejudicando, saúde, mental, dos, gregos}

Tabela 2: Coleção de artigos

No exemplo, a palavra "Grécia" aparece nos documentos, (2), (3) e (4), assim seu suporte é 3; a palavra "crise" aparece em (3) e (5), logo seu suporte é 2; a palavra "de" ocorre em todos os documentos, exceto em (4) e (5), assim seu suporte é 3; a palavra "europeia" ocorre em (1), (3) e (4), logo seu suporte é 3. Todas as outras palavras ocorrem apenas em um único documento. Logo se for definido o limiar de suporte $l = 2$, teremos como conjuntos de itens frequentes os conjuntos {Grécia}, {crise}, {de} e {europeia}. Além disso, as duplas {Grécia, europeia}, {Grécia, de} e {europeia, de} ocorrem em 2 documentos, logo também são conjuntos de itens frequentes.

No caso do algoritmo FIHC, conjuntos de itens frequentes são também denominados de conjuntos globais de itens frequentes, e suporte é denominado de *global support*.

3.1.1 Algoritmo Apriori

A extração dos conjuntos de itens frequentes é uma das técnicas mais utilizadas para caracterização de dados com objetivo de resumir os atributos gerais mais relevantes de um conjunto de dados. O exemplo mostrado na seção anterior é uma forma de encontrar os conjuntos de itens frequentes, porém esse método não é eficiente quando precisa ser utilizado em uma coleção grande de documentos. Um dos algoritmos mais populares para tal propósito, seja em um banco de dados ou em documentos textuais, é o algoritmo Apriori [12].

O algoritmo utiliza a propriedade de anti-monotonicidade [10], onde se um conjunto de itens não é frequente, então todos os seus superconjuntos também não serão.

Ele inicia a busca a partir de uma coleção de conjunto de itens frequentes com cardinalidade $k = 1$, chamado de F_1 . A partir daí todos os conjuntos

de cardinalidades maiores são obtidos a partir de uma coleção de candidatos C. O algoritmo para quando não encontrar mais nenhum conjunto de itens frequentes.

Algorithm 1 Algoritmo Apriori para extração de conjuntos de itens frequentes em documentos textuais

1: $F_1 \leftarrow$ (Conjunto de itens frequentes de cardinalidade 1)
2: $k \leftarrow 1$
3: **while** $F_k \neq \emptyset$ **do**
4: $\quad C_{k+1} \leftarrow candidate\_gen(F_k)$
5: $\quad$ **for all** documento d na coleção **do**
6: $\quad\quad C'_d \leftarrow subset(C_{k+1}, d)$
7: $\quad\quad$ **for all** candidato c $\in C'_d$ **do**
8: $\quad\quad\quad c.count \leftarrow c.count + 1$
9: $\quad F_{k+1} \leftarrow \{c \in C_{k+1} | c.count \geq$ suporte mínimo$\}$
10: $\quad k \leftarrow k + 1$
$\quad$ **return** $\cup_k F_k$

3.2 O que é *cluster frequent item* e *cluster support*

Dado o grupo C, um elemento de um conjunto global de itens frequentes é denominado de *cluster frequent item* de C se este elemento está contido em um número mínimo de documentos de C, definido como *minimum cluster support*.

Além disso, *cluster support* de um item em C é definido como a porcentagem de documentos de C que contém o item.

3.3 O que é tf-idf

Tf-idf é uma medida estatística usada para avaliar o quão importante uma palavra é para um documento em relação a uma coleção de documentos [10]. A ideia é que se uma palavra é muito comum em diferentes documentos da coleção, então provavelmente ela tem pouca relevância, como por exemplo, a conjunção aditiva "e". A importância aumenta proporcionalmente ao número de vezes que uma palavra aparece no documento, mas é compensado pela frequência da palavra no corpus. O tf-idf é dado pelo produto de duas medidas: a frequência do termo e a frequência inversa do documento, dado por:

$$\text{tf-idf}(t, d) = \text{tf}(t, d) \times \log \frac{N}{\text{df}(t)} \tag{5}$$

onde $\text{tf}(t, d)$ é a razão entre o número de vezes que o termo t aparece no documento d e a quantidade de termos em d; N é a quantidade de documentos na coleção e $\text{df}(t)$ é a quantidade de documentos da coleção que possui o termo t. Existem diversas variantes dessa medida como visto em [4].

3.4 Construção dos grupos

Apresentado alguns conceitos relacionados ao FIHC, vamos ao algoritmo propriamente dito. Como já mencionado, o primeiro passo do algoritmo é construir os grupos, e isso é feito em duas etapas. Primeiramente, os grupos iniciais são construídos a partir dos conjuntos globais de itens frequentes extraídos da coleção pelo algoritmo Apriori e então é feito o processo de disjunção desses grupos para que, ao final, um documento pertença somente a um único grupo.

Na criação dos grupos iniciais, para cada conjunto global de itens frequentes é criado um grupo que inicialmente incluirá todos os documentos que contêm em seu corpo esse conjunto. A esse conjunto de itens frequentes é dado o nome de conjunto de itens frequentes gerador do grupo.

Note que esses grupos podem não ser disjuntos, pois um documento pode conter vários conjuntos de itens frequentes.

Documento	mercado	crise	universo	Grécia	inteligência
artigo.1	0	0	3	0	2
artigo.2	2	0	0	1	0
artigo.3	1	4	0	3	0
artigo.4	0	0	4	0	1
artigo.5	0	1	0	0	2
artigo.6	3	2	0	0	0
artigo.7	1	0	0	2	2
artigo.8	0	1	0	0	0
artigo.9	0	0	2	0	1
artigo.10	0	0	3	0	0

Tabela 3: Exemplo de uma coleção de documentos e os respectivos valores tf-idf de cada palavra em relação a cada documento

Conjunto global de itens frequentes	*Global support*
{mercado}	40%
{crise}	40%
{universo}	40%
{Grécia}	30%
{inteligência}	50%
{mercado, Grécia}	30%
{universo, inteligência}	30%

Tabela 4: Conjuto global de itens frequentes da coleção de exemplo da tabela 3. O *minimum global support* é definido como 30%. A relação completa dos conjuntos de itens da coleção pode ser encontrada no anexo A.

3.4.1 Disjunção dos grupos sobrepostos

A disjunção é um processo importante, pois garante que ao final do algoritmo cada documento pertença somente ao grupo que melhor o descreva. Para cada documento D, é listado todos os grupos no qual ele pertence, e após isso é calculado para cada um desses grupo C uma medida que indicará a "relevância" de D em relação a C. Essa medida tem o nome de Score e é proposto em [7].

Após os cálculos da medida Score de D para cada grupo C, é mantido o documento somente no grupo que maximizou essa medida. Esse cálculo de relevância de um grupo inicial C_i para um documento D_j é definido por:

$$\begin{aligned} Score(C_i \leftarrow D_j) = &|\sum_{x} tf - idf(x, D_j) * cluster\_support(x)| \\ &-|\sum_{x'} tf - idf(x', D_j) * global\_support(x')| \end{aligned} \tag{6}$$

onde x representa um *global frequent item* que está contido em D_j e também é um *cluster frequent item* de C_i; x' representa um *global frequent item* que está contido em D_j, mas que não é um *cluster frequent item* de C_i; $tf - idf(x, D_j)$ e $tf - idf(x', D_j)$ são as medidas tf-idf dos itens x e x' em D_j; $cluster\_support(x)$ é o *cluster support* de x e $global\_support(x')$ é o *global support* de x'.

O primeiro termo da função $Score(C_i \leftarrow D_j)$ recompensa C_i se o *global frequent item* x em D_j é também um *cluster frequent item* de C_i e o segundo termo penaliza C_i se o *global frequent item* x' em D_j não é um *cluster frequent item* de C_i. De acordo com Martin Ester *et al.* [7], intuitivamente, um grupo C_i é "bom" para um documento D_j se há muitos *global frequent items* em D_j

Grupo	Documentos do grupo	*Cluster frequent items* e *cluster support*
C(mercado)	artigo.2, artigo.3, artigo.6 e artigo.7	(mercado/cs = 100%), (Grécia/cs = 75%), (crise/cs = 50%)
C(crise)	artigo.3, artigo.5, artigo.6 e artigo.8	(crise/cs = 100%), (mercado/cs = 100%)
C(universo)	artigo.1, artigo.4, artigo.9 e artigo.10	(universo/cs = 100%), (inteligência/cs = 75%)
C(Grécia)	artigo.2, artigo.3 e artigo.7	(Grécia/cs = 100%), (mercado/cs = 100%)
C(inteligência)	artigo.1, artigo.4, artigo.5, artigo.7 e artigo.9	(inteligência/cs = 100%), (universo/cs = 60%)
C(mercado, Grécia)	artigo.2, artigo.3 e artigo.7	(mercado/cs = 100%), (Grécia/cs = 100%)
C(universo, inteligência)	artigo.1, artigo.4 e artigo.9	(universo/cs = 100%), (inteligência/cs = 100%)

Tabela 5: Grupos criados a partir dos conjuntos globais de itens frequentes da tabela 4, onde o *minimum cluster support* = 50%.

que aparecem em "muitos" documentos em C_i. Se há mais de um grupo que maximiza a medida $Score(C_i \leftarrow D_j)$, então é escolhido o grupo com maior número de itens em seu conjunto de itens frequentes gerador.

Seguindo o exemplo da tabela 5, note que o documento artigo.3 pertence aos grupos C(mercado), C(crise), C(Grécia) e C(mercado, crise). Para se decidir em qual desses grupos o artigo deve permanecer, são calculados os seguintes *Score*:

$Score(C(mercado) \leftarrow artigo.3) = 1 * 1 + 4 * 0,5 + 3 * 0,75 = 5,25$
$Score(C(crise) \leftarrow artigo.3) = 1 * 0,5 + 4 * 1 - 3 * 0,3 = 3,6$
$Score(C(Grécia) \leftarrow artigo.3) = 1 * 1 + 3 * 1 - 4 * 0,4 = 2,4$
$Score(C(mercado, Grécia) \leftarrow artigo.3) = 1 * 1 + 3 * 1 - 4 * 0,4 = 2,4$

Desta maneira, o documento artigo.3 seria mantido apenas no grupo C(mercado).

3.5 Criação da árvore de grupos

Uma vez computado os grupos iniciais, podemos visualizar cada um deles como um tópico ou subtópico na coleção de documentos. Agora é necessário

Grupo	Documentos do grupo	*Cluster frequent items* e *cluster support*
C(mercado)	artigo.3 e artigo.6	(mercado/cs = 100%), (Grécia/cs = 75%), (crise/cs = 50%)
C(crise)	artigo.8	(crise/cs = 100%), (mercado/cs = 100%)
C(universo)	artigo.10	(universo/cs = 100%), (inteligência/cs = 75%)
C(Grécia)		nenhum
C(inteligência)	artigo.5	(inteligência/cs = 100%), (universo/cs = 60%)
C(mercado, Grécia)	artigo.2 e artigo.7	(mercado/cs = 100%), (Grécia/cs = 100%)
C(universo, inteligência)	artigo.1, artigo.4 e artigo.9	(universo/cs = 100%), (inteligência/cs = 100%)

Tabela 6: Grupos disjuntos

criar a árvore, colocando os tópicos mais generalistas no topo e os mais específicos abaixo. Essa árvore tem como objetivos criar uma estrutura para a navegação entre os documentos e também facilitar a poda dos grupos muito específicos.

Os grupos são inicialmente aninhados na árvore de acordo com a quantidade de itens em seu conjunto global gerador. Dessa forma, aqueles que possuem somente um item, estão no primeiro nível da árvore, e assim por diante. A raiz (nível 0) da árvore é por definição um grupo vazio e sem nenhum item em seu conjunto global de itens frequentes. Como dito no início desse capítulo, o FIHC utiliza a abordagem aglomerativa para a criação da hierarquia, ou seja, para cada grupo C no nível k é escolhido um pai dentre os potenciais grupos do nível *k-1*, cujo o conjunto global seja subconjunto dos conjuntos globais de itens frequentes de C. Se houver mais de um potencial pai no nível *k-1*, então é escolhido aquele grupo que maximiza a medida *Score*, similar ao processo descrito na seção 3.4.1. Isso é melhor detalhado em [7].

Uma vez escolhido um grupo pai para cada grupo da coleção, é necessário verificar se é possível unificar os grupos similares ou com tópicos muito específicos, garantindo uma hierarquia mais natural para a navegação e melhorando a acurácia do algoritmo. Isso é feito de duas formas: 1) através do processo de poda dos grupos filhos e 2) unificação dos grupos similares do

nível 1.

A medida de similaridade utilizada no algoritmo FIHC utiliza a ideia de tratar um grupo como um documento conceitual, definindo a partir da combinação de todos os documentos do grupo, e a partir daí é tirado a medida através de um cálculo bem similar à medida *Score* apresentada na seção anterior. Dado um grupo C_a e C_b, a medida de similaridade entre os grupos é calculado a partir da média geométrica entre a similaridade de C_a para C_b e da similaridade de C_b para C_a. Da seguinte forma:

$$Inter\_Sim(C_a \leftrightarrow C_b) = [Sim(C_a \leftarrow C_b) * Sim(C_b \leftarrow C_a)]^{\frac{1}{2}} \tag{7}$$

onde a similaridade Sim de C_a para C_b é dado como:

$$Sim(C_a \leftarrow C_b) = \frac{Score(C_a \leftarrow doc(C_b))}{\sum_x \text{tf-idf}(x, doc(C_b)) + \sum_{x'} \text{tf-idf}(x', doc(C_b))} + 1 \tag{8}$$

Em $Sim(C_i \leftarrow C_j)$, a medida *Score* é a mesma discutida na seção anterior, porém a única diferença é que o documento $doc(C_j)$ utilizado no cálculo é um documento conceitual, construído a partir da combinação de todos os documentos da subárvore de C_j, e tf-idf$(x, doc(C_b))$ e tf-idf$(x', doc(C_b))$ são, respectivamente, as medidas tf-idf dos itens x e x' em $doc(C_j)$.

3.6 Rotulagem do grupo

Uma etapa importante no processo de agrupamento é a de rotulagem dos grupos, pois para que um usuário entenda o resultado do algoritmo e o utilize sem dificuldades é necessário que a rotulagem seja legível e faça sentido a ele.

No caso do FIHC, a própria abordagem de criação dos grupos a partir dos conjuntos globais de itens frequentes da coleção e posterior escolha dos *cluster frequent items* torna natural pensar que esses conjuntos de palavras também são bons candidatos para dar nome aos próprios grupos.

4 Arquitetura da biblioteca e sistema hVINA

A biblioteca proposta neste trabalho tem como objetivo tratar todo os passos de uma solução para agrupamento, desde o pré-processamento dos documentos até a implementação do algoritmo de agrupamento de fato. Inicialmente foi implementado o algoritmo FIHC, porém sua arquitetura é modular e permite a substituição por outros algoritmos.

Já o sistema tem como objetivo apresentar o agrupamento dos artigos jornalísticos de forma simples e intuitiva para os usuários, onde as configurações dos parâmetros do algoritmo FIHC seja pouco necessária por eles, uma vez que a crescente utilização da internet como meio de divulgação de artigos pelos jornais e revistas torna necessário um mecanismo automático e não-supervisionado para extração e organização do conhecimento, buscando encontrar entre os artigos relacionamentos a princípio escondidos. A figura 5 mostra o processo para o agrupamento dos artigos.

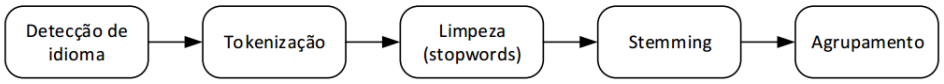

Figura 5: Sequência de passos da biblioteca proposta.

A biblioteca detecta e trata artigos em português, inglês e espanhol, porém permite a extensão para outros idiomas, bastando escolher um algoritmo para tokenização e criar uma lista de *stopwords* específicos do idioma. Além disso, a biblioteca é modular, permitindo a troca dos algoritmos utilizados em cada processo de forma simples e sem a necessidade de uma mudança geral no sistema.

Os processos de detecção de idioma, tokenização, remoção dos *stopwords* e *stemming* utilizam técnicas e algoritmos já bem conhecidos e empregados na área de reconhecimento de padrões.

O sistema hVINA foi desenvolvido utilizando o *framework* Rails [16] e outras tecnologias Web, como a marcação HTML, Javascript e CSS3. Todo o código-fonte segue junto a esta monografia.

4.1 Detecção de idioma

O processo de detecção do idioma da coleção de documentos é parte essencial em um sistema de agrupamento, uma vez que algoritmos empregados nos

passos seguintes, como o de tokenização e remoção dos *stopwords*, requerem o conhecimento do idioma utilizado nos documentos.

4.1.1 Método de n-grama

A tarefa de encontrar o idioma de um documento é em si própria um problema na área de reconhecimento de padrão, visto que é preciso classificar o documento em um conjunto de classes (idiomas). Um algoritmo bastante conhecido para essa tarefa baseia-se nas frequências dos n-gramas do documento. Esse algoritmo utiliza o processo de aprendizagem supervisionado, pois cria o modelo dos n-gramas mais frequentes de cada idioma a partir de um conjunto de textos de treinamento.

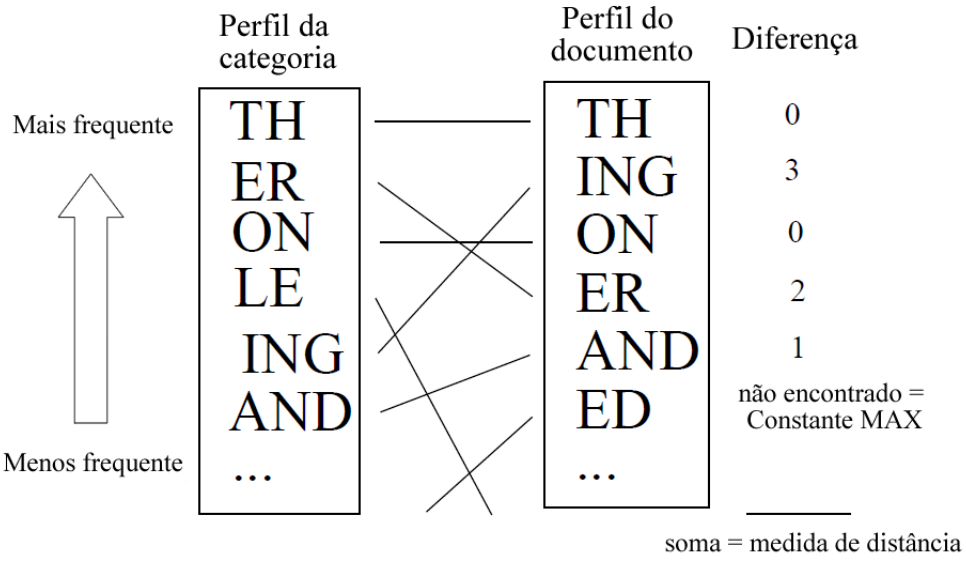

Figura 6: Esquema do método n-grama, retirado de [11]. O perfil à esquerda representa o modelo n-grama do idioma inglês, obtido a partir de um conjunto de textos para treinamento, o perfil à direita representa os n-gramas mais frequentes de um documento de exemplo e a última coluna indica a distância entre a posição desses n-grama em relação ao perfil do idioma.

4.2 Tokenização

Como um dos pontos centrais do algoritmo FIHC é a utilização dos conjuntos de itens frequentes da coleção de documentos, uma ação importante para

que a biblioteca funcione com boa acurácia é estabelecer um processo que encontre nos textos todos os termos que o compõe. Isso se dá pelo processo de tokenização ou segmentação de texto. De acordo com Manning *et al.* [4], tokenização é o processo de quebra do documento em partes, chamados de *tokens*. A definição de *token* varia de acordo com o contexto do problema, e na prática pode ser palavras, frases ou símbolos. No caso deste trabalho, os *tokens* são encontrados à medida que um espaço em branco ou uma marcação gráfica aparece entre eles, exceto o hífen. É necessário falar também que neste trabalho as palavras *token*, termo e item se equivalem. Veja a figura 7 para um exemplo ilustrativo desse processo:

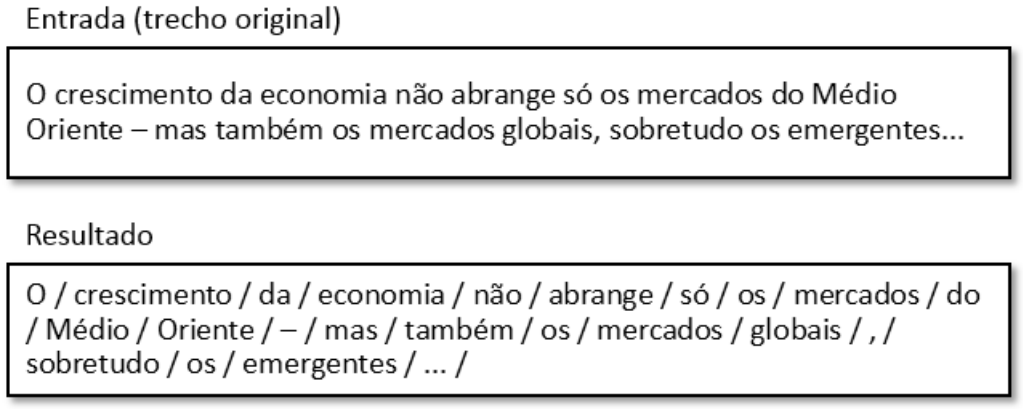

Figura 7: Exemplo do processo de tokenização em um trecho de artigo.

Um bom algoritmo para extração dos tokens deve levar em consideração o idioma do texto, pois em geral o processo de tokenização é difícil, devido a construções de palavras que podem surgir durante o processo, como as contrações de palavras inglesas, como *"we've many time (...)"*, ou as palavras compostas em português como "latino-americano".

O algoritmo de tokenização implementado inicialmente na biblioteca é o *Treebank tokenizer* [13], porém a extensão para outros algoritmos de tokenização é possível devido à modularidade da biblioteca.

4.3 Limpeza

Um atributo requerido para um sistema de agrupamento de documentos é a escolha de grupos (tópicos) significativos, porém nem sempre o algoritmo de agrupamento sozinho é capaz de determinar a representatividade de uma determinada palavra no momento da construção dos grupos. Por esse motivo,

é usado a técnica de limpeza para eliminação das palavras que são extremamente comuns no idioma desses documentos. Essas palavras são chamadas de *stopwords*. Uma estratégia simples para determinar a lista delas é ordená-las pela frequência que aparecem na coleção, porém geralmente essa lista é composta pelas palavras de classes gramaticais como preposição, artigo, advérbio, numeral, pronome e pontuações em geral.

No caso deste trabalho é feito também a eliminação das marcações gráficas que não fazem parte de uma palavra propriamente dita.

Continuando com o exemplo da figura 8, com a remoção dos *stopwords* o resultado seria da seguinte forma:

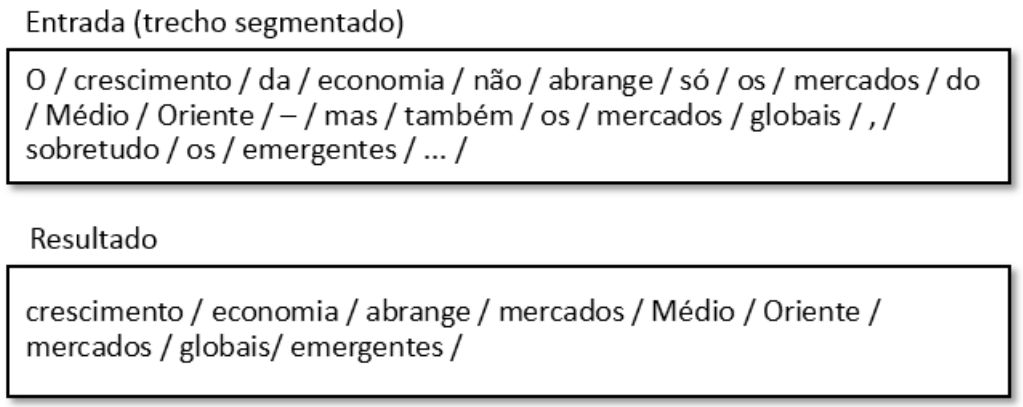

Figura 8: Exemplo do processo de limpeza no conjunto de palavras resultante na figura 7.

4.4 *Stemming*

Stemming é o processo de unificar as formas variantes de uma palavra em uma representação comum, chamado de *stem* [5]. Por exemplo, as palavras "corredor", "corrida" e "correria" podem ser reduzidas para a representação "corr".

Esse processo é essencial para um sistema de agrupamento, pois, assim como o processo de limpeza, tem o objetivo de aumentar a eficiência do algoritmo de agrupamento.

4.4.1 Tipos de erros

Geralmente os diversos algoritmos de *stemming* são analisados através de dois tipos de erros:

- *Overstemming*: quando é retirado da palavra um sufixo grande o suficiente para que palavras com significados diferentes sejam unificadas numa única representação. Por exemplo, quando as palavras "escritor" e "escrutínio" são reduzidas para "escr".

- *Understemming*: quando é retirado da palavra um sufixo suficientemente pequeno para que palavras com o mesmo significado sejam separadas em representações distintas. Por exemplo, quando as palavras "alimentar" e "alimentação" são reduzidas para "alimenta" e "alimentaç", respectivamente.

4.4.2 O algoritmo RSLP

A maioria dos algoritmos para *stemming* baseia-se no método de remoção de afixos (sufixos e prefixos), definidos a partir de um conjunto de regras predefinidas. Caso a palavra contenha o prefixo ou sufixo definido por alguma regra, ele é removido [6].

O problema desse tipo de método é que esses algoritmos são muito dependentes da linguagem para os quais foram criados, assim é difícil a criação de um algoritmo universal.

No caso da língua inglesa, o algoritmo mais usado é o de Porter [4], já no caso da língua portuguesa o algoritmo escolhido para este estudo é o Removedor de Sufixos da Língua Portuguesa (RSLP), devido ao seu bom desempenho [5].

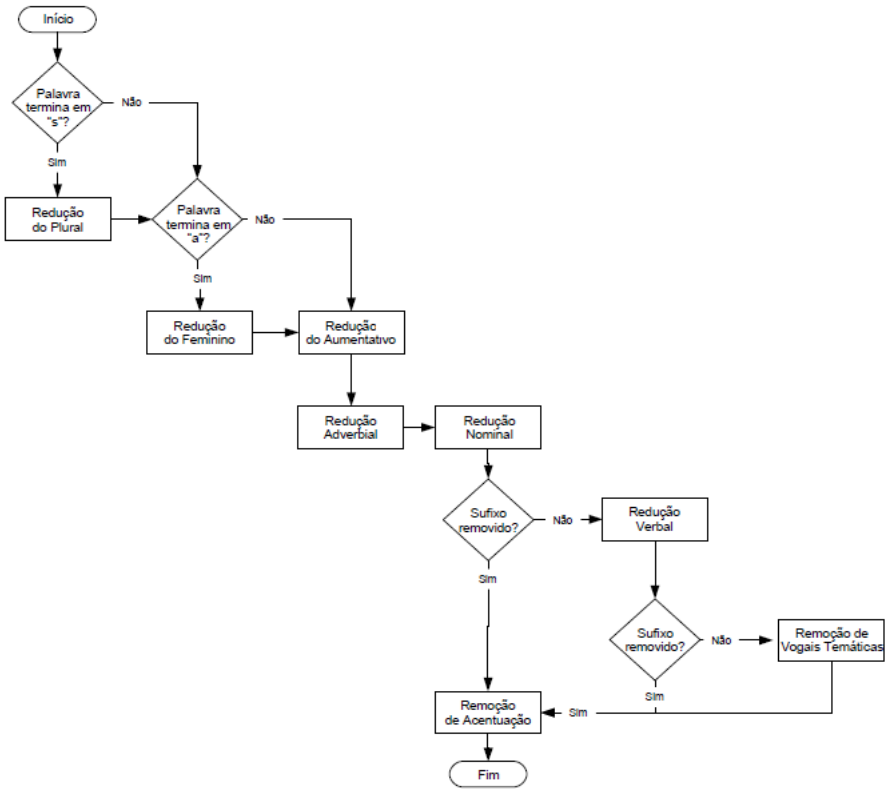

Figura 9: Sequência de passos do algoritmo RSLP, retirado de [5].

Um exemplo de regra utilizada no algoritmo RSLP é:

$$\text{“is”, 2, “il”, \{ “lápis", “cais”, “mais”, “crúcis”, “pois”, “dois”, “leis” \}} \qquad (9)$$

Essa regra é utilizada para redução de plural nos casos onde a palavra termina com o sufixo “is”. Assim “is” é o sufixo a ser substituído, 2 é o tamanho mínimo para o *stem* final, “il” é o novo valor do sufixo, e as palavras “lápis”, “cais”, “mais”, “crúcis”, “pois”, “dois” e “leis” são as exceções da regra.

4.5 Agrupamento

Até então os processos discorridos neste capítulo fazem parte da etapa de pré-processamento da coleção de documentos. Aqui começa de fato o processo principal da biblioteca, que é o de agrupar os artigos jornalísticos. Como visto na seção 2.4.2, existem diversos algoritmos para agrupamento hierárquico desenvolvidos na área de análise de agrupamento.

No primeiro momento foi implementado o algoritmo FIHC, estudado no capítulo 3. Porém será visto na próxima seção que a implementação de outros algoritmos é simples.

4.6 Interface gráfica

Com o intuito de demostrar a utilização da biblioteca, esse trabalho também inclui a criação do sistema hVINA, acrônimo para *Hierarchical Viewer of News Articles*. Com esse sistema é possível avaliar a viabilidade da biblioteca de agrupamento de artigos jornalísticos com alguns dos maiores portais de publicações do meio digital. A interface inicial do sistema está ilustrado pela figura 10 de onde é possível escolher alguma das publicações pré-selecionadas, ou então importar uma coleção de artigos jornalísticos de interesse do usuário.

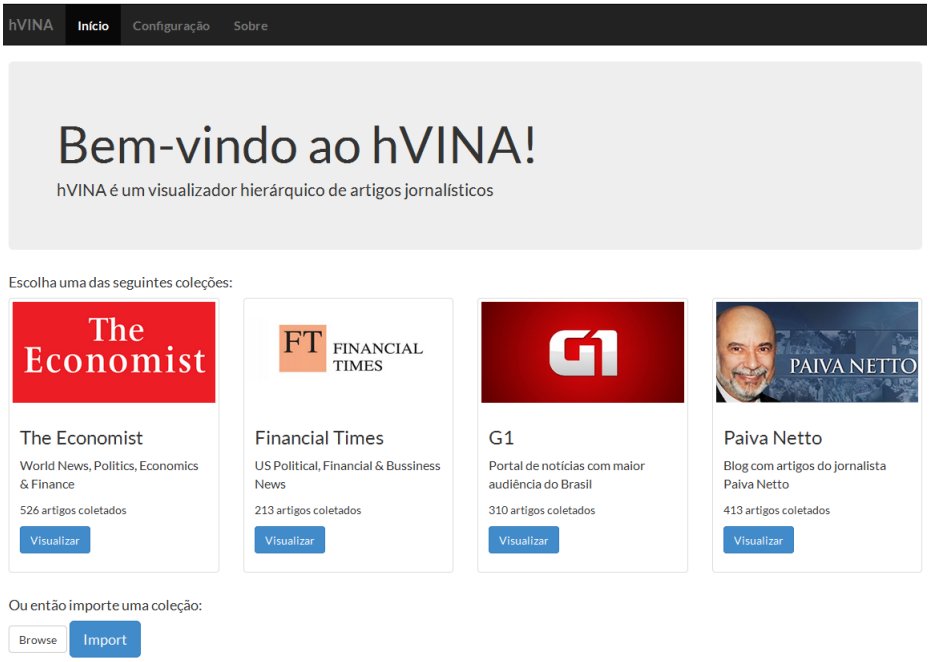

Figura 10: Tela inicial do sistema hVINA

O hVINA também permite a configuração do algoritmo de agrupamento, no caso em especial o FIHC, de forma simples.

Figura 11: Tela de configuração do algoritmo de agrupamento.

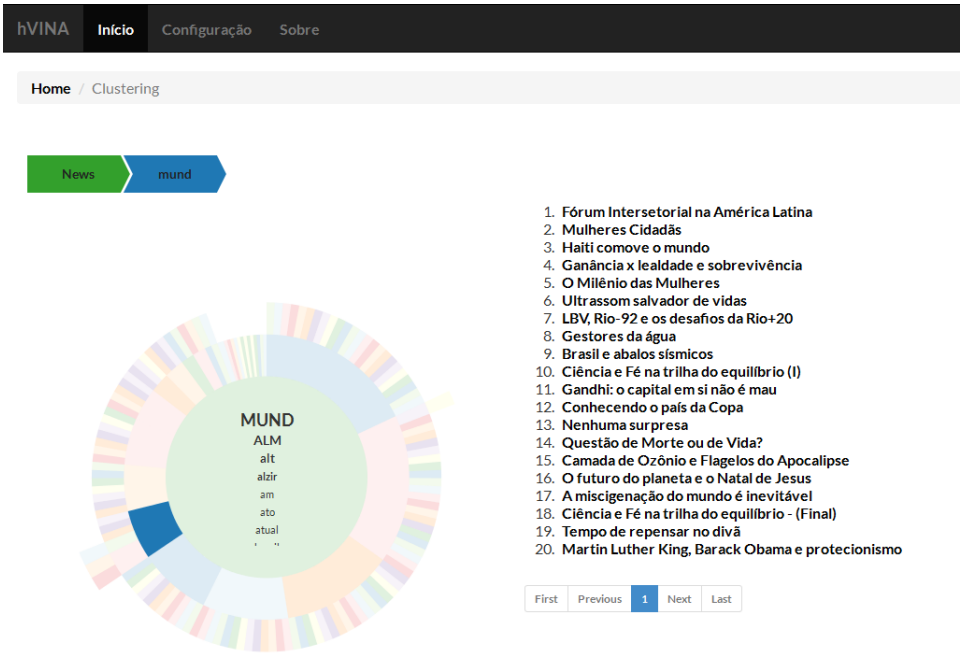

Figura 12: Tela para visualização do agrupamento.

5 Resultados

5.1 A biblioteca

A biblioteca foi desenhada utilizando alguns padrões de projeto, dentre eles o *Strategy Pattern* para encapsular os algoritmos de agrupamentos em classes independentes e o *Template Method Pattern* para auxiliar na definição das classes desses algoritmos numa estrutura comum, permitindo que novos algoritmos sejam implementados ou substituídos sem a necessidade de alteração na estrutura da biblioteca.

Desde sua concepção, ela teve como requisito ser modular, o que permitirá a inclusão de outros algoritmos de agrupamento de forma simples e sem a necessidade de alteração dos módulos e classes principais da biblioteca. O projeto está disponível no repositório online Github, no endereço https://github.com/tomferreira/hierarchical_clustering.

A biblioteca foi desenvolvida em Ruby e segue a especificação do *RubyGem* e por esse motivo pode ser facilmente reutilizada em outros projetos Ruby. Sua principal classe é a HierarchicalClustering, de onde é controlado o fluxo de dados entre os passos da biblioteca. A figura 13 ilustra a utilização da biblioteca, adotando o algoritmo FIHC para o agrupamento e imprimindo o resultado num formato aceito pelo sistema hVINA.

```
fihc = Clustering::Fihc::Controller.new(
    global_support: 0.20, cluster_support: 0.30, k_clusters: 30)
fihc.output_manager(HVinaOutputManager.new("results"))

hc = HierarchicalClustering.new( dir: "./artigos", algorithm: fihc )
```

Figura 13: Exemplo de utilização da biblioteca com o algoritmo FIHC.

Outro ponto importante é que a coleção dos artigos deve seguir um padrão e conter alguns dados obrigatórios como o título, conteúdo e URL do artigo. Cada artigo deve estar separado em arquivo diferentes e os dados dispostos no formato JSON como mostrado no exemplo da figura 14.

```
{
  “title”:  “A revolução que falta acontecer”,
  “content”: “Há cerca de dez anos, li na Tribuna da Imprensa (...)”,
  “link”: “http://www.paivanetto.com.br/index.php/pt/art/83379”
}
```

Figura 14: Exemplo de um documento aceito pela biblioteca.

A formatação do resultado do agrupamento também pode ser estendida, bastando criar uma classe que deverá herdar a classe Clustering::Fihc::OutputManager e informá-la ao algoritmo utilizado. A figura 15 ilustra um exemplo.

```
class HVinaOutputManager < Clustering::Fihc::OutputManager
  # Implementação suprimida
end

(...)
fihc.output_manager(VinaOutputManager.new("results"))
```

Figura 15: Exemplo de extensão da formatação do resultado de um agrupamento.

5.2 O sistema hVINA

Apesar do intuito inicial do sistema hVINA ser um protótipo para a demonstração prática da biblioteca desenvolvida, ele é totalmente funcional e pode ter mais funcionalidades acrescentadas.

O sistema foi desenvolvido em Ruby juntamente com o *framework* Rails, e seu código-fonte também está disponível no repositório Github, no endereço https://github.com/tomferreira/hVINA.

5.3 Testes

Nesta seção, são analisados e avaliados os resultados obtidos em cada etapa da biblioteca de agrupamento de artigos jornalísticos.

5.3.1 Tempo de processamento

Para avaliar a eficiência da biblioteca, foi utilizado um conjunto de 413 artigos do escritor e jornalista José de Paiva Netto [15], todos escritos em português.

Foram avaliados o tempo médio de execução em cada etapa e a quantidade de tokens encontrados. Como parâmetros do algoritmo FIHC foram utilizados os seguintes valores sugeridos pelos próprios autores do algoritmo [7] e que também são usados por padrão no sistema hVINA: *global support* = 0,25; *cluster support* = 0,3 e k-grupos = 30.

O resultado da tabela 7 mostra o tempo médio a partir de 10 (dez) execuções[1] da biblioteca, tendo como entrada o conjunto de artigos descrito acima.

	Deteção	Tokenização	Limpeza	*Stemming*	Agrupamento	Total
Tempo médio	6,57	0,56	2,65	7,61	84,17	101,57
Desvio padrão	0,55	0,01	0,07	0,32	1,61	1,15

Tabela 7: Tempo médio de execução, em segundos, de cada etapa da biblioteca.

Esse resultado revela que o maior tempo de processamento gasto pela biblioteca está no agrupamento propriamente dito, sendo cerca de 83% do tempo total.

5.3.2 Tokenização e limpeza

As etapas de tokenização e limpeza são mostradas na tabela 8, que apresenta a quantidade de tokens encontrados pelo algoritmo de tokenização e, desse conjunto, a quantidade removida na etapa de limpeza (remoção dos *stopwords*). A coluna de redução representa proporcionalmente a quantidade de tokens restantes em relação ao conjunto inicial encontrado.

[1] Os testes foram feitos com o Ruby MRI 2.2.0 e Suse Linux 11 quad-core de 3,2 GHz.

	Encontrados	Removidos	Restantes	Redução
Tokens	337.706	201.098	136.608	59,5 %
Tokens distintos	29.773	5.614	24.159	18,8 %

Tabela 8: Resultados das etapas de tokenização e limpeza.

6 Conclusão

Esse trabalho discutiu a análise de agrupamento de documentos textuais com ênfase em coleções de artigos jornalísticos, além de estudar algumas classes de algoritmos que podem ser usadas para tal propósito.

O resultado observado nos testes, demonstra que ainda é necessário melhorar a eficiência de tempo da biblioteca, principalmente na etapa de agrupamento em si (algoritmo FIHC).

O objetivo principal foi fazer essa análise a partir da criação de uma biblioteca que abrangesse todas as etapas de uma solução para agrupamento, desde o pré-processamento da coleção até a execução do algoritmo de agrupamento em si, tendo como requisito o fato de que não fosse necessário nenhum conhecimento prévio da coleção por parte do usuário.

Inicialmente foi implementado na biblioteca o algoritmo de agrupamento *Frequent Itemset-based Hierarchical Clustering* (FIHC), visto que o agrupamento hierárquico é uma forma mais natural para a navegação nos grupos (tópicos) e permite reconhecer relações implícitas entre esses grupos de maneira muito mais simples.

Além da biblioteca, foi desenvolvido também o sistema *Hierarchical Viewer of News Articles* (hVINA) para ser uma interface gráfica dela, permitindo a escolha da coleção e a visualização do agrupamento resultante de forma muito mais intuitiva aos usuários finais.

O resultado obtido pela biblioteca em conjunto com o sistema hVINA demostra a viabilidade de uma solução que tenha boa usabilidade e seja amigável a qualquer usuário - mesmo aqueles não técnicos, onde não haja a necessidade de treinamento do algoritmo e que sua configuração seja quase zero.

6.1 Considerações finais

Segundo o professor Clay Shirky, da New York University, o problema da *overdose* de informação, da qual estamos sujeitos diariamente, está no fato de não sabermos filtrar o que é importante para nós [17]. Logo, uma possível solução é justamente criar mecanismos automáticos que possam organizar essas informações de modo a nos ajudar a decidir o que é importante ou não

para nós.

Agrupar grande quantidade de artigos de forma que os grupos sejam intuitivos ao usuário, isto é, que possuam rótulos significativos, não é uma tarefa fácil, visto que a diversidade de tópicos possíveis é imensa. A solução apresentada neste trabalho é somente uma das possíveis abordagens, e novas tentativas de resolver esse problema devem também ser empenhadas com outras técnicas.

6.2 Análise subjetiva

6.2.1 Desafios e frustrações encontradas

Durante o início do desenvolvimento do trabalho, houveram diversos desafios quanto à definição do que seria feito. A área de reconhecimento de padrão é, em particular, muito empolgante e com imensas aplicações práticas. Porém, encontrar algum problema que fosse, ao mesmo tempo, capaz de ser debruçado e "solucionado" em apenas 1 ano (menos ainda se considerarmos que tive que realizar diversas tarefas não propriamente ditas de análise ou desenvolvimento) e também útil e relevante, não foi simples.

Uma vez fechado o escopo - e dado um título mais detalhado do meu trabalho, as demais tarefas começaram a caminhar bem.

Conciliar esse trabalho com outras matérias do curso e demandas de outra ordem, como emprego e vida pessoal, não foi nada simples. Mas agora no final vejo que assim como o conhecimento técnico adquirido na área de reconhecimento de padrão, adquiri também um grande amadurecimento quanto à organização de todas essas tarefas.

Já nas questões técnicas, houveram algumas frustrações durante o transcorrer do trabalho, como a falta de material acadêmico mais farto de algoritmos que tratasse em específico de agrupamento de documentos textuais.

Infelizmente só foi implementado um único algoritmo de agrupamento - o FIHC. Outros algoritmos estavam em vista, porém alguns problemas na implementação do FIHC adiaram as demais implementações.

O maior problema técnico do projeto no entanto é o desempenho do algoritmo FIHC, que ainda precisaria de muito trabalho para chegar a um tempo aceitável (quase tempo real).

6.2.2 Disciplinas relevantes para o trabalho

As disciplinas mais relevantes durante o desenvolvimento deste trabalho foram:

1. **Armazenamento e Recuperação de Informação (MAC0333):** Essa disciplina introduziu os principais conceitos da área de recuperação de informação, como tf-idf e medida F.

2. **Engenharia de software (MAC0332):** Durante todo o processo de desenvolvimento e gerenciamento das atividades envolvidas no processo de criação da biblioteca e do sistema foram empregados técnicas apresentadas nesta disciplina.

3. **Programação orientada a objeto (MAC0441) e Laboratório de programação II (MAC0242):** Tanto a biblioteca quanto o sistema utilizam o paradigma de programação orientada à objeto. Em especial foi em Laboratório de programação que pela primeira vez tive em contato com o desenvolvimento de um grande projeto de software.

4. **Estrutura de dados (MAC0323):** Muitas das estruturas de dados do algoritmos de agrupamento foram baseados ou emprestados dos conceitos apresentado nesta disciplina.

5. **Introdução à computação (MAC0110) e Princípios de Desenvolvimento de Algoritmos (MAC0122):** Essas disciplinas foram a base necessária para um bom aprendizado de todo o conceito teórico e todas as outras disciplinas.

6.2.3 Trabalhos futuros

Melhorar o desempenho do algoritmo de agrupamento FIHC é um trabalho que pode ser continuado, e técnicas como a programação distribuída podem contribuir para essa melhoria.

Além disso, implementar a detecção de outros idiomas na biblioteca é algo que naturalmente precisará ser feito. E novos gráficos para a visualização desses grupos podem ser utilizados para complementar o já existente, enriquecendo ainda mais a experiência do usuário com o sistema.

Outra possibilidade é integrar a biblioteca ao projeto DocumentCloud (http://www.documentcloud.org/public/search/).

6.2.4 Agradecimentos

Agradeço a todos os professores do curso de Ciências da Computação, em especial ao Alair Pereira do Lago que muito me ajudou durante o desenvolvimento desse trabalho, fazendo sempre comentários muito valiosos para mim. Por fim, agradeço aos meus pais e minha irmã por todo apoio e paciência durante o ano todo.

7 Anexo A

Conjunto de itens	*Global support*
{mercado}	40%
{crise}	40%
{universo}	40%
{Grécia}	30%
{inteligência}	50%
{mercado, crise}	20%
{mercado, universo}	0%
{mercado, Grécia}	30%
{mercado, inteligência}	10%
{crise, universo}	0%
{crise, Grécia}	10%
{crise, inteligência}	10%
{universo, Grécia}	0%
{universo, inteligência}	30%
{Grécia, inteligência}	10%
{mercado, crise, universo}	0%
{mercado, crise, Grécia}	10%
{mercado, crise, inteligência}	0%
{mercado, universo, Grécia}	0%
{mercado, universo, inteligência}	0%
{mercado, Grécia, inteligência}	10%
{crise, universo, Grécia}	0%
{crise, universo, inteligência}	0%
{crise, Grécia, inteligência}	0%
{universo, Grécia, inteligência}	0%
{mercado, crise, universo, Grécia}	0%
{mercado, crise, universo, inteligência}	0%
{mercado, crise, Grécia, inteligência}	0%
{mercado, universo, Grécia, inteligência}	0%
{crise, universo, Grécia, inteligência}	0%
{mercado, crise, universo, Grécia, inteligência}	0%

Tabela 9: Relação completa dos conjuntos de itens da coleção de documentos da tabela 3

Referências

[1] http://redarterj.wordpress.com/2009/10/15/a-primeira-biblioteca-do-mundo

[2] Sergios Theodoridis, Konstantinos Koutroumbas. *Pattern Recognition*. Elsevier; 3 edition, 2006.

[3] Jain, A.K., Murty, M.N., Flynn, P.J. *Data Clustering: A Review*. ACM Computing Surveys 31 (3), 264–322, 1999.

[4] Manning, Christopher D.; Raghavan, Prabhakar; Schütze, Hinrich. *An Introduction to Information Retrieval*. Cambridge: Cambridge University Press, 569p, 2009.

[5] Viviane Orengo, Cristian Huyck. *A Stemming Algorithm for the Portuguese Language*. Proceedings of the Eighth International Symposium on String Processing and Information Retrieval, 2001.

[6] Alexandre Ramos Coelho. *Stemming para a lingua portuguesa: Estudo, análise e melhoria do algoritmo RSLP*. 2007.

[7] Benjamim C. M. Fung, Ke Wang, Martin Ester. *Hierarchical Document Clustering Using Frequent Itemsets*. 2003.

[8] Toby Segaran. *Programming collective intelligence : building smart Web 2.0 applications*. O'Reilly, 1 edition, 2008.

[9] John Wang. *Encyclopedia of Data Warehousing and Mining*. IGI Global Snippet, 2 edition, 2009.

[10] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 2 edition, 2011.

[11] William B. Cavnar, John M. Trenkle. *N-Gram-Based Text Categorization*. In Proceedings of SDAIR-94, 3rd Annual Symposium on Document Analysis and Information Retrieval, 1994.

[12] R. Agrawal and R. Srikant. *Fast algorithms for mining association rules in large databases*. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pages 487-499, Santiago, Chile, September, 1994.

[13] http://www.nltk.org/_modules/nltk/tokenize/treebank.html

[14] http://www.dormantroot.com/2014/02/my-experiment-with-clustering.html

[15] http://www.paivanetto.com.br/index.php/pt/artigos

[16] http://rubyonrails.org/

[17] Web 2.0 Expo NY: Clay Shirky. It's Not Information Overload. It's Filter Failure. http://blip.tv/web2expo/web-2-0-expo-ny-clay-shirky-shirky-com-it-s-not-information-overload-it-s-filter-failure-1283699