Classificação não-supervisionada hierárquica de artigos jornalísticos

Cirillo Ribeiro Ferreira

20 de Outubro de 2014

Universidade de São Paulo

Instituto de Matemática e Estatística

MAC 499 - Trabalho de Formatura Supervisionado

Supervisor: Alair Pereira do Lago

Sumário

I Parte objetiva

1. Introdução
   - 1.1. Motivação
   - 1.2. Objetivos
   - 1.3. Organização do trabalho

2. Fundamentos
   - 2.1. Reconhecimento de padrão
     - 2.1.1. Tipos de processo de aprendizagem
   - 2.2. Análise de agrupamento
   - 2.3. Agrupamento de documentos textuais
   - 2.4. Formas de agrupamentos textuais
     - 2.4.1. Agrupamento plano
     - 2.4.2. Agrupamento hierárquico
   - 2.5. Critérios de avaliação

3. FIHC
   - 3.1. O que é *Frequent Itemset*
     - 3.1.1. Algoritmo Apriori
   - 3.2. O que é *cluster frequent item* e *cluster support*
   - 3.3. O que é tf-idf
   - 3.4. Construção dos clusters
     - 3.4.1. Disjunção dos clusters sobrepostos
   - 3.5. Criação da árvore de clusters
   - 3.6. Rotulagem do cluster

4. Arquitetura do sistema de agrupamento
   - 4.1. Detecção de idioma
     - 4.1.1. Método *n-gram*
   - 4.2. Tokenização
   - 4.3. Limpeza
   - *4.4. Stemming*
     - 4.4.1. Tipos de erros
     - 4.4.2. O algoritmo RSLP

4.5. Agrupamento [Ver nome melhor para a seção]
4.6. Interface gráfica

5. Resultados
   5.1. A biblioteca
   5.2. O sistema hVINA
   5.3. Avaliação do agrupamento
6. Conclusão

II Parte subjetiva

1. Desafios e frustrações encontradas
2. Disciplinas relevantes para o trabalho
3. Trabalhos futuros
4. Agradecimentos

III Referências bibliográficas

PARTE I
OBJETIVA

Capítulo 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. Desde então, profissões como bibliotecário, documentalista e arquivista foram criadas para atuar na organização de todo esses documentos produzidos pela humanidade.

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

A área de classificação de documentos é bem interessante e possui diversas aplicações práticas como classificação de spam, identificação do idioma utilizado e análise de sentimento, porém, em especial, artigos jornalísticos têm um enorme desafio devido à grande quantidade de novos documentos gerados 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 falta de tempo para uma leitura mais crítica e ciente de um contexto maior devido a enorme quantidade de notícias que estamos sujeitos a receber todos os dias e da organização simplória feita pelos grandes portais de notícias.

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 sim, 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 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 clusters, pois facilita a utilização pelos usuários.

1.2 Objetivos

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

1. Estudos das principais classes de algoritmos para agrupamento de documentos textuais.
2. Criação de uma biblioteca para agrupamento de artigos jornalísticos.
3. Proposta e implementação do sistema hVINA (*Hierarchical Viewer of News Articles*) para análise de agrupamento de artigos jornalísticos.

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

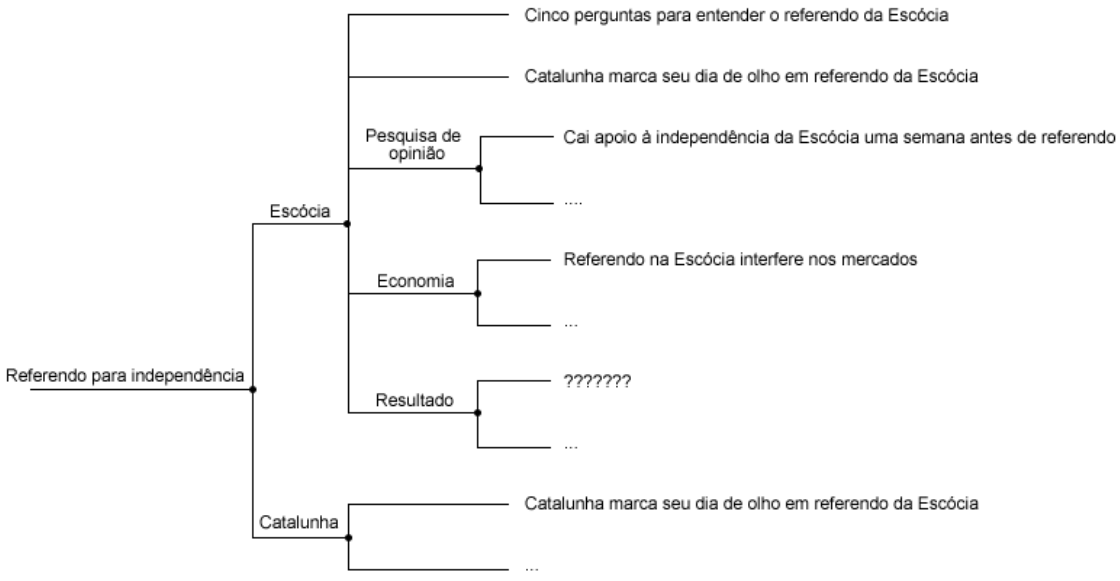

Figura ?: “Esquema do resultado esperado”

1.3 Organização do trabalho

No capítulo 2 é apresentado os fundamentos da área de reconhecimento de padrão e mais especificamente de análise de agrupamento. No capítulo 3 é discutido em detalhe o algoritmo FIHC, escolhido para ser implementado no sistema proposto no 4o capítulo para agrupamento de artigos jornalísticos. No capítulo 5, é feito a avaliação do sistema na coleção de artigos do jornalista José de Paiva Netto e também da coleção da Reuters ...

Capítulo 2

Fundamentos

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

2.1 Reconhecimento de padrão

Segundo Theodoridis e Koutroumbas [1], 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 permeia todas as áreas de atuação humana, principalmente numa sociedade pós-industrial, onde a necessidade de automação das diversas atividades é essencial. Sua aplicação vai deste sistemas que utilizam visão computacional, passando por reconhecimento de caracteres até sistemas para auxílio à domada 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 denonimado de conjunto de treinamento (*training set*). Porém, em muitos casos a criação de um conjunto de treinamento é inviável devido a 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. [1?, 2]

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 cujo membros sejam similares entre si e diferentes dos membros de outros grupos [2]. 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 [1], 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 ecologia), tipologia (em ciências sociais) e partição (em teoria dos grafos).

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 de características, que envolve em sintetizar os dados em características mais relevantes para o problema principal sem 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 utilizam 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 casos, 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 [1, 2], porém neste capítulo serão citados duas categorias de algoritmos mais empregados nesse processo. Uma explicação com aspectos mais detalhados desses métodos pode ser encontrado em [1, 2].

2.4.1 Agrupamento plano

Agrupamento plano é a categoria de algoritmos cujo os clusters resultantes são dados num único nível. O principal algoritmo dessa categoria é o *k-means*.

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

O agoritmo inicia com uma escolha aleatória dos *k* centróides e a cada iteração é feito um refinamento na escolha desses centróides a fim de …. até que … . A figura ? exemplifica a sequência descrita acima.

Há duas propriedades importantes do *k-means*:

1. Cada objeto deve pertencer ao menos um dos *k* clusters.
2. Nenhum objeto pertence a mais que um cluster.

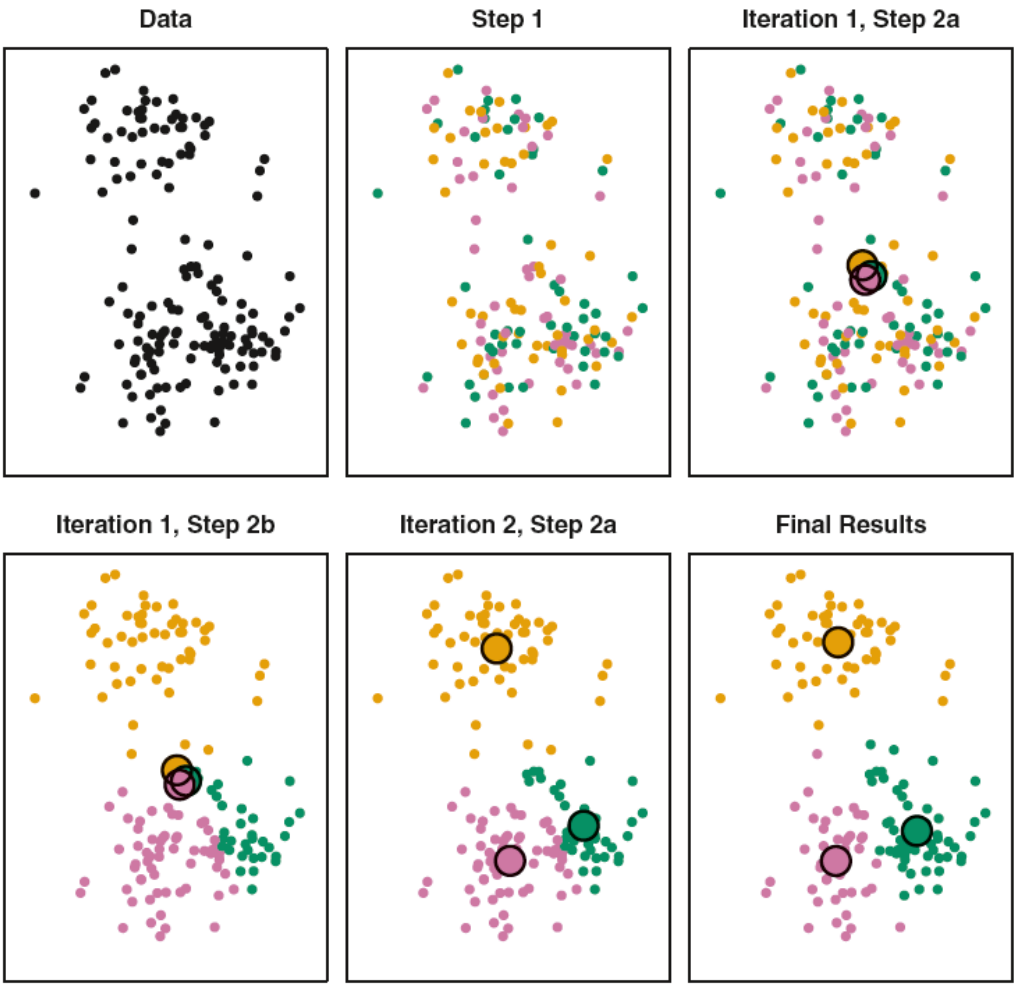

Figura ?: Sequência de passos do algoritmo *k-means.* [colocar referência]

Um cuidado especial na utilização desse algoritmo é na escolha do número de clusters desejado. Uma escolha inadequada poderá gerar um mau agrupamento, unindo duas classes naturais em um único cluster ou dividindo uma classe natural em dois clusters, porém algumas técnicas tem sido desenvolvidas para resolver esse problema [2, ?]. Um exemplo de problema que pode ocorrer está ilustrado pela figura ?.

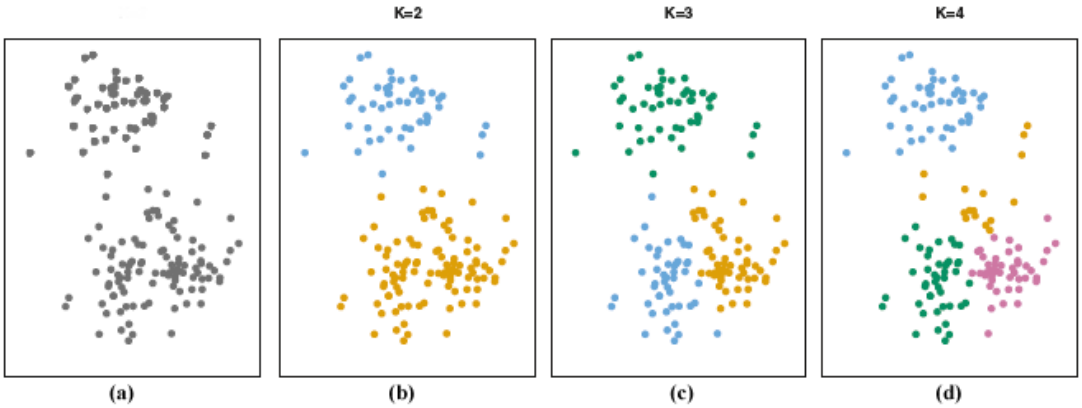

Figura ?: Em (a) tem um conjunto de dados fictício com três classes naturais. Em (b) quanto é escolhido o número de clusters k = 2, as classes naturais 2 e 3 são unidas num único cluster. Em (c), com k = 3, os clusters refletem as classes naturais dos dados, e em (d) com k = 4 a classe natural 3 é dividida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

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 cluster na entrada e o fato de não serem determinísticos [3]. Dessa forma, algoritmos que criassem agrupamentos multiníveis e que tivessem como requisito opcional a entrada do número de clusters foram desenvolvidos. Esses algoritmos criam uma hierarquia de clusters, uma estrutura que gera mais informação …
Existem duas abordagem no funcionamento de um algoritmo hierárquico que são:

Abordagem aglomerativa:. Gera a árvore de cluster ao continuamente calcular a similaridade de todos os pares de clusters 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 ?.

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

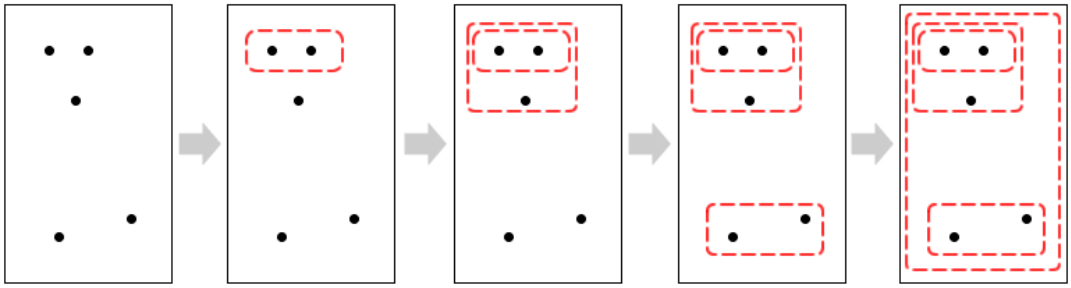

Figura ?: Exemplo de um algoritmo hierárquico aglomerativo, retirado de [8], onde o critério de similaridade utilizado é a distância euclidiana.

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 cluster, e o segundo, chamado de falso negativo (FN) ou erro tipo II, ocorre quando o algoritmo associa dois documentos similares a clusters distintos. Um critério de avaliação justo tem de levar em conta as ocorrências desses dois erros. A tabela ? a seguir, mostra a relação entre ...

	Mesmo Cluster	Diferentes clusters
Documentos similares	Verdadeiro positivo (TP)	Falso negativo (FN) ou Erro tipo II
Documentos não similares	Falso positivo (FP) ou Erro tipo I	Verdadeiro negativo (TN)

Tabela ?:

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

$$P = \frac{TP}{TP + FP}$$

Já a sensibilidade mede a fração de documentos que …. e é dado pela fórmula:

$$R = \frac{TP}{TP + FN}$$

Por exemplo, 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 cluster C = {1, 3, 4, 5} a precisão de C em relação a N_1 é P = 3/4 e a sensibilidade 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 sensibilidade, ou vice-versa? A resposta para essa pergunta depende do contexto onde é aplicado o agrupamento. Segundo Manning *et al*, na maiorida das vezes é mais aceitável o erro de tipo I, pois separar documentos similares em clusters distintos é geralmente pior do que juntar documentos não similares num mesmo cluster [7].

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

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

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

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

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

Já definindo $\beta > 1$, a precisão será mais enfatizada em relação à sensitividade...

Note que para a F-measure, o algoritmo não basta ter apenas boa precisão ou apenas boa sensibilidade, 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 F-measure se enquadra na segunda categoria, pois avalia o agrupamento a partir de uma classificação de referência [7].

Capítulo 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 (*frequent itemset*).

Segundo Martin Ester *et al.*, a ideia utilizada para o critério de agrupamento dos documentos é que existe algum *frequent itemset* para cada cluster (tópico) no conjunto de documentos, e diferentes clusters compartilham poucos *frequent itemset*.

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 clusters, 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 clusters desejado é opcional, assim o usuário não precisa ter o conhecimento prévio da coleção de documentos.

O algoritmo utiliza a abordagem aglomerativa, descrita na seção 2.4.2, e possui dois passos principais, que é a construção dos clusters e posteriormente a criação da árvore hierárquica, porém antes de apresentá-los é necessário introduzir algumas definições.

3.1 O que é *Frequent Itemset*

Por definição, *itemset* é o conjunto de itens que ocorrem em conjunto na coleção de documentos. Logo *frequent itemset* é o conjunto de itens que ocorrem numa quantidade de documentos acima de um limiar de apoio definido.

Em outras palavras, assumindo que *l* é o limiar de apoio. Se F é um conjunto de itens, o apoio (*support* em inglês) de F é a proporção de documentos no qual F é um subconjunto dos itens desses documentos. Assim, dizemos que F é um *frequent itemset* se o seu valor de apoio é *l* ou superior. A figura ? ilustra um exemplo:

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

No exemplo acima, a palavra "Grécia" aparece em todos artigos, exceto em (1) e (5), 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 artigos, 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 artigo. Logo se for definido o limiar de apoio $l = 2$, teremos como *frequent itemset* 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 *frequent itemsets.*

No caso do algoritmo FIHC, *frequent itemsets* são também denominados de *global frequent itemsets*, e *support* é denominado de *global support*.

3.1.1 Algoritmo Apriori

A extração dos *frequent itemsets* é uma das técnicas mais utilizadas para caracterização de dados com objetivo de resumir os atributos gerais mais relevantes de um conjunto de dados. Um dos algoritmo mais populares para tal propósito, seja em um banco de dados ou em documentos textuais, é o algoritmo Apriori. (wikipedia?)

A algoritmo utiliza a propriedade de anti-monotonicidade [10], onde se um *itemset* não é frequente, então todo o seu superconjunto também não será.

Ele inicia a busca a partir de um conjunto de *frequent itemsets* inicial com cardinalidade k = 1, chamado de F_1. A partir daí todos os *frequent itemsets* de cardinalidades maiores são obtidos a partir de um conjunto de candidatos C. O algoritmo para quando não encontrar mais nenhum *frequent itemsets*. A figura ? mostra o algoritmo:

F_1 = (Frequent itemsets de cardinalidade 1);
Para (k = 1; F_k != ⊘ ; k++) faça
    C_{k+1} = candidade-gen(F_k);
    Para cada documento d na coleção faça
        C'_d = subset(C_{k+1}, d);
        Para cada candidato c E C'_d faça
            c->count <- c->count + 1
        F_{k+1} = {c E C'_d / c->count >= suporte mínimo}

Figura ?: Algoritmo Apriori para extração em documentos textuais

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

Dado o cluster C_i, um item de um *global frequent itemset* é denominado de *cluster frequent item* de C_i se este item está contido em um número mínimo de documentos de C_i.

Além disso, *cluster support* de um item em C_i é definido como a porcentagem de documentos de C_i 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 uma coleção de documentos. (wikipedia) A ideia é que se uma palavra é tão comum em diferentes documentos da coleção, então provavelmente ela tem pouca relevância, 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:

$$tf - idf(t,d) = tf(t,d) \times log\ N/df(t)$$

onde *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 *df(t)* é a quantidade de documentos da coleção que possui o termo *t*. Existem diversas variantes dessa medida como visto em [3].

3.4 Construção dos clusters

Apresentado alguns conceitos relacionados ao FIHC, vamos ao algoritmo propriamente dito. Como já dito, o primeiro passo do algoritmo é contruir os clusters, e isso é feito em dois passos. Primeiramente, os clusters iniciais são contruidos a partir dos *global frequent itemsets* computados na coleção e então é feito o processo de disjunção desses clusters para que no final um documento pertença somente a um único cluster.

Na criação dos clusters iniciais, para cada *global frequent itemset* é criado um cluster que inicialmente incluirá todos os documentos que contem esse *itemset* em seu corpo. Logo cada cluster possuirá um *global frequent itemset* gerador.

Note que esses clusters podem não ser disjuntos, pois um documento pode conter vários *frequent itemsets*.

3.4.1 Disjunção dos clusters sobrepostos

A disjunção é um processo importante, pois garante que cada documento pertença somente ao caminho mais significativo da hierarquia final. Para cada documento e clusters no qual ele pertence, é calculado uma medida *Score*, e então é mantido o documento somente no cluster que maximiza essa medida. A medida *Score* de um cluster inicial C_i para um documento doc_j é definida por:

$$Score(Ci \leftarrow docj) = |\Sigma\, tfidf(x,\ docj) * cluster\_support(x)| - |\Sigma\, tfidf(x',\ docj) * global\_support(x')|$$

onde x representa um *global frequent item* que está contido em doc_j e também é um *cluster frequent item* de C_i; x' representa um *global frequent item* que está contido em doc_j, mas que não é um *cluster frequent item* de C_i; tfidf(x, docj) e tfidf(x', docj) são as medidas tf-idf dos itens x e x' em doc_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(Ci ← docj)* recompensa C_i se o *global frequent item* x em doc_j é também um *cluster frequent item* de C_i e o segundo termo penaliza C_i se o *global frequent item* x' em doc_j não é um *cluster frequent item* de C_i. De acordo com Martin Ester *et al.* [7], intuitivamente, um cluster C_i é "bom" para um documento doc_j se há muitos *global frequent items* em doc_j que aparecem em "muitos" documentos em C_i. Se há mais de um cluster que maximiza a medida *Score(Ci ← docj)*, então é escolhido o cluster com maior número de itens em seu *global frequent itemset* gerador.

3.5 Criação da árvore de clusters

Uma vez computado os clusters iniciais, podemos visualizar cada um deles como um tópico ou subtópico na coleção de documentos. Agora é necessário criar a árvore, colocando os tópicos mais generalistas no topo e os mais específicos abaixo. Essa árvore tem como objetivos cria uma estrutura para a navegação entre os documentos e também facilitar a poda dos clusters muito específicos.

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

Uma vez escolhido um cluster pai para cada cluster da coleção, é necessário verificar se é possível unificar os clusters 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 clusters filhos e 2) unificação dos cluster similares do nível 1.

A medida de similaridade utilizada no algoritmo FIHC, utiliza a ideia de tratar um cluster como um documento conceitual, definindo a partir da combinação de todos os documentos do cluster, e a partir daí e tirado a medida através de um cálculo bem similar à medida *Score* apresentada na seção anterior. Dado um cluster C_a e C_b, a medida de similaridade entre os clusters 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:

$$Inter\_Sim(Ca \leftrightarrow Cb) = [Sim(Ca \leftarrow Cb) * Sim(Cb \leftarrow Ca)]^{\frac{1}{2}}$$

Onde a similaridade *Sim de* Ci para Cj é dado como:

$$Sim(Ci \leftarrow Cj) = \frac{Score(Ci \leftarrow doc(Cj))}{\sum_{x} tfidf(x, doc(Cj)) + \sum_{x'} tfidf(x', doc(Cj))} + 1$$

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 Cj, e tfidf(x, doc(C_j)) e tfidf(x', doc(C_j)) são, respectivamente, as medidas tf-idf dos itens x e x' em doc_j.

3.6 Rotulagem do cluster

…

Capítulo 4

Arquitetura do sistema de agrupamento de artigos jornalísticos

A biblioteca proposta neste trabalho tem como objetivo implementar o algoritmo FIHC para agrupamento de artigos jornalísticos, porém desenhado preparado para implementações futuras de outros algoritmos.

Já o sistema tem como objetivo apresentar o agrupamento dos artigos jornalísticos que 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 ? mostra o processo para o agrupamento dos artigos.

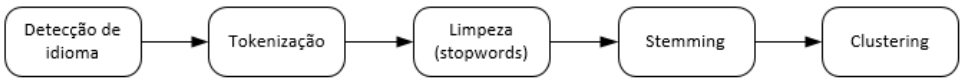

Figura ?: Sequência de passos da biblioteca proposta

A biblioteca trabalha com artigos em inglês e português, porém foi desenvolvida preparada para facilitar a extensão de 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 processos 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 [?] e outras tecnologias Web, como Javascript e CSS3. As implementações seguem 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 de documentos, uma vez que algoritmos empregados nos passos seguintes, como o de tokenização e remoção dos *stopwords*, necessitam do conhecimento prévio do idioma dos documentos.

4.1.1 Método de *n-gram*

Em si a tarefa de encontrar o idioma de um documento é um problema de reconhecimento de padrão. Na prática é preciso classificar o documento em uma classes pré-estabelecidas (idiomas). Um método bastante conhecido para essa classificação é o baseado nas frequências dos n-gram do documentos.

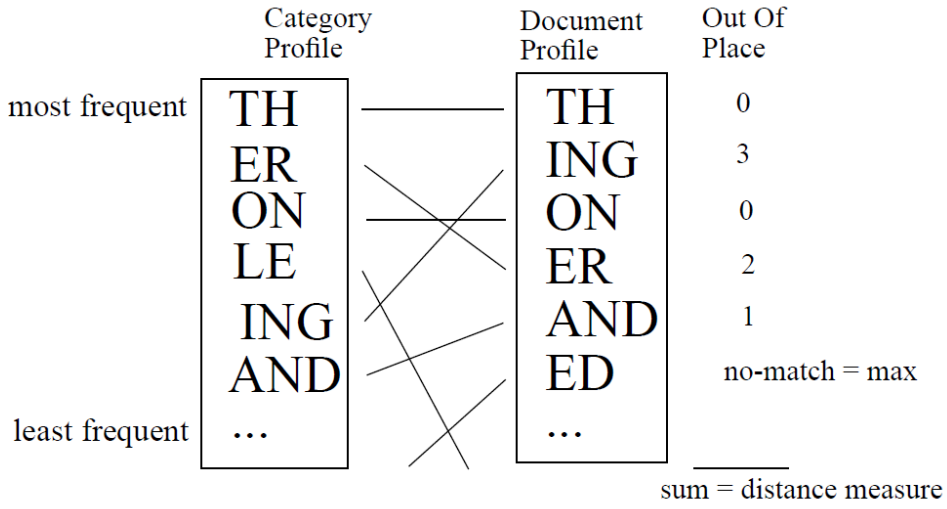

Figura ?: Esquema do método n-gram, retirado de [11]

4.2 Tokenização

Como um dos pontos centrais do algoritmo FIHC é a utilização dos *frequent itemsets* da coleção de documentos, uma ação importante para que o sistema funcione com boa acurácia é estabelecer um processo que encontre nos textos todos os termos que o compõe, eliminando, no caso deste trabalho, marcações gráficas que não fazem parte de uma palavra propriamente dita. Isso se dá pelo processo de tokenização ou segmentação de texto. De acordo com Manning *et al.* [3], 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 o esquema ? para um exemplo ilustrativo desse processo:

Entrada:

Capitu, apesar daqueles olhos que o diabo lhe deu... Você já reparou nos olhos dela? São assim de cigana oblíqua e dissimulada.

Saída:

Capitu / apesar / daqueles / olhos / que / o / diabo / lhe / deu / Você / já / reparou / nos / olhos / dela / São / assim / de / cigana / oblíqua / e / dissimulada

Figura ?:

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 de podem surgir, como as contrações de palavras inglesas, como “we’ve many time...”, ou as palavras compostas em português como “beija-flor”.

4.3 Limpeza

Um atributo necessário para um sistema de agrupamento de documentos é a escolha de clusters (tópicos) significativos e que é feito pelo algoritmo de agrupamento, porém nem sempre o algoritmo sozinho é capaz de determinar a representatividade de uma determinada palavra no momento da construção dos clusters. Por esse motivo, é usado uma técnica de pré-processamento de documentos 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 quantidade de vezes 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.

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

Capitu / apesar / daqueles / olhos / diabo / deu / Você / já / reparou / nos / olhos / São / assim / cigana / oblíqua / dissimulada

4.4 *Stemming*

Stemming é o processo de unificar as formas variantes de uma palavra em uma representação comum, chamado de *stem* [4]. 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 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 palavras 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 [5].

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 [citacao], 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 [4].

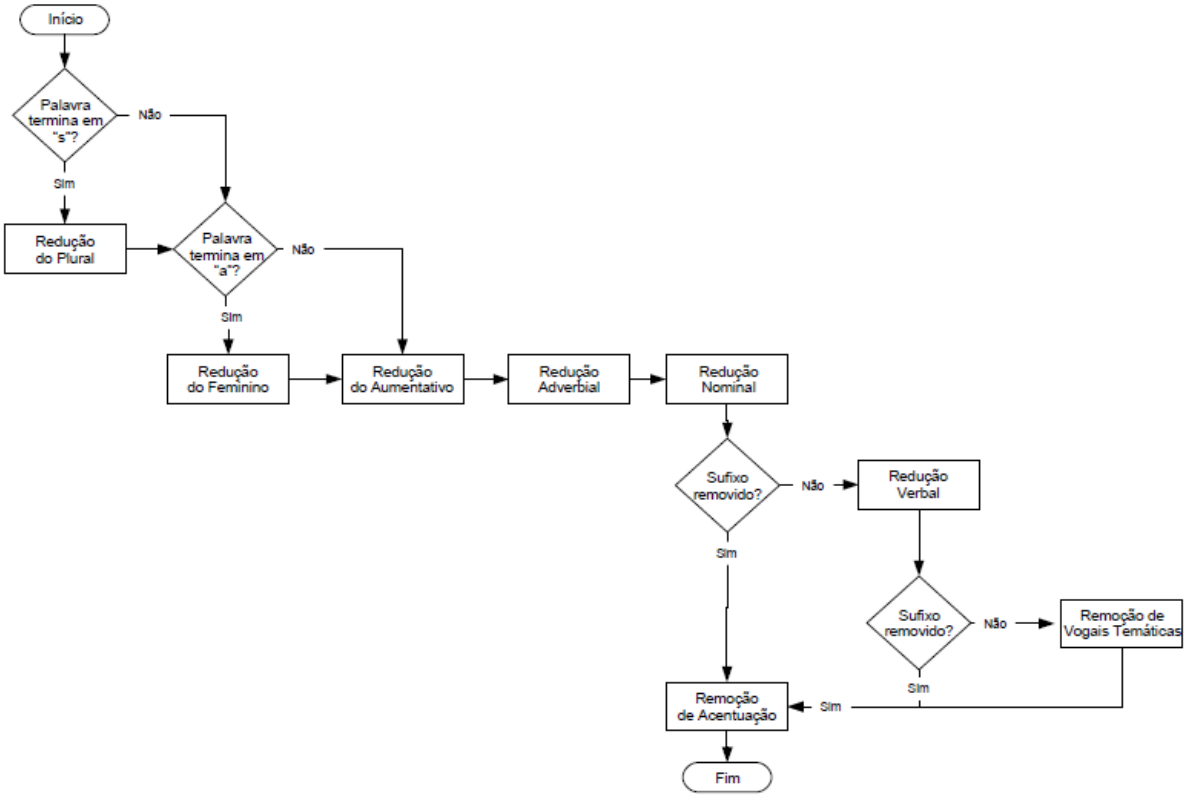

Figura ?: Sequência de passos do algoritmo RSLP, retirado de [4].

Um exemplo de regra utilizada no algoritmo RSLP é:

“is", 2, “il”, { “lápis”, “cais”, “mais”, “crúcis”, “pois”, “dois”, “leis” }

Essa regra é utilizada para redução de pural 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

No processo de agrupamento dos artigos é utilizado o algoritmo FIHC, explicado no capítulo 3…

4.7 Interface gráfica

….

5. Resultados

5.1 A biblioteca

….

5.2 O sistema hVINA

….

5.3 Avaliação do agrupamento

Para avaliação do sistema foi utilizado uma coleção de artigos do jornalista José de Paiva Netto [?]. Uma característica importante dessa coleção é o jornalista em muitas ocasiões aborda mais de um tema num único artigo, pois como eles são produzidos semanalmente, na maioria das vezes o jornalista inclui, após a abordagem do tema principal da semana, agradecimentos ou relato de algum evento importante.

Por causa desse fato, resolvi também avaliar a eficiência do sistema numa coleção baseada na coleção original, porém com a remoção desses assuntos secundários.

...

6. Conclusão

….

PARTE II
SUBJETIVA

1. Desafios e frustrações encontradas

…

2. Disciplinas relevantes para o trabalho

...

3. Trabalhos futuros

...

III Referências bibliográficas

[1] Sergios Theodoridis, Konstantinos Koutroumbas. Pattern Recognition. 3 ed. [S.l.]: Elsevier, 2006.

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

[3] MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich. An Introduction to Information Retrieval. Cambridge: Cambridge University Press, 569p, 2009.

[4] Viviane Orengo, Cristian Huyck. A Stemming Algorithm for the Portuguese Language.

[5] Alexandre Ramos Coelho. Stemming para a lingua portuguesa: estudo, análise e melhoria do algoritmo RSLP.

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

[8] Segaran, Toby. Programming collective intelligence : building smart Web 2.0 applications.

[9] John Wang. Encyclopedia of Data Warehousing and Mining. 556p, ??

[10] Jure Leskovec, Anand Rajaraman, Jeffrey D. Ullman. Mining of Massive Dataasets. 2 ed. Cambridge University Press.
*(*http://infolab.stanford.edu/~ullman/mmds/ch6.pdf)

[11] William B. Cavnar, John M. Trenkle. N-Gram-Based Text Categorization. (http://odur.let.rug.nl/vannoord/TextCat/textcat.pdf)

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

[?] http://rubyonrails.org/

[?] http://www.paivanetto.com.br