UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DESENVOLVIMENTO DE UM ALGORITMO PARA CLASSIFICAÇÃO DE TEXTOS DE ATAS DO COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO DA UFSM EM TÓPICOS HIERÁRQUICOS

TRABALHO DE GRADUAÇÃO

Tháygoro Minuzzi Leopoldino

Santa Maria, RS, Brasil

2016

DESENVOLVIMENTO DE UM ALGORITMO PARA CLASSIFICAÇÃO DE TEXTOS DE ATAS DO COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO DA UFSM EM TÓPICOS HIERÁRQUICOS

Tháygoro Minuzzi Leopoldino

Trabalho de Graduação apresentado ao Curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Bacharel em Ciência da Computação

Orientador: Prof. Dr. Sergio Luis Sardi Mergen

Trabalho Nº 427
Santa Maria, RS, Brasil

2016

Minuzzi Leopoldino, Tháygoro

Desenvolvimento de um algoritmo para classificação de textos de atas do colegiado de Ciência da Computação da UFSM em tópicos hierárquicos / por Tháygoro Minuzzi Leopoldino. – 2016.
41 f.: il.; 30 cm.

Orientador: Sergio Luis Sardi Mergen
Monografia (Graduação) - Universidade Federal de Santa Maria, Centro de Tecnologia, Curso de Ciência da Computação, RS, 2016.

1. Hierarquização. 2. Sumarização. 3. Ata. I. Sardi Mergen, Sergio Luis. II. Título.

E-mail: thaygoro@inf.ufsm.br

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

DESENVOLVIMENTO DE UM ALGORITMO PARA CLASSIFICAÇÃO DE TEXTOS DE ATAS DO COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO DA UFSM EM TÓPICOS HIERÁRQUICOS

elaborado por
Tháygoro Minuzzi Leopoldino

como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

COMISSÃO EXAMINADORA:

Sergio Luis Sardi Mergen, Dr.
(Presidente/Orientador)

Ana Trindade Winck, Dr. (UFSM)

Giovani Rubert Librelotto, Dr. (UFSM)

Santa Maria, 16 de Dezembro de 2016.

AGRADECIMENTOS

Primeiramente agradeço a UFSM por ter proporcionado a estrutura e espaço para que eu pudesse estudar e crescer. Agradeço também a todos os professores que contribuíram para minha formação, em especial aos professores Giovani Librolotto e Ana Winck que formaram a banca deste trabalho, e ao professor Sérgio Mergen, que teve bastante paciência e compreensão ao me orientar neste projeto.

Agradeço também a todos os colegas de curso, em especial ao Thiago Trugillo, Otávio Rodrigues, Alberto Kummer e Emmanuel Katende, que me acompanharam em inúmeros trabalhos, estudos e desafios, e os quais considero amigos para a vida toda.

Agradeço enormemente a minha família, em especial ao meu pai que me apresentou o mundo da computação e da programação, e claro, a minha amada mãe que sempre me apoiou e me deu forças, independente das minhas falhas.

A todos aqueles que não foram citados mas fizeram parte da minha tragetória de alguma forma, meu muito obrigado.

“Lembrar que você vai morrer é a melhor maneira que eu conheço para evitar a armadilha de pensar que você tem algo a perder. Você está nú. Não há razão para não seguir seu coração.”

— STEVE JOBS

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

DESENVOLVIMENTO DE UM ALGORITMO PARA CLASSIFICAÇÃO DE TEXTOS DE ATAS DO COLEGIADO DE CIÊNCIA DA COMPUTAÇÃO DA UFSM EM TÓPICOS HIERÁRQUICOS

AUTOR: THÁYGORO MINUZZI LEOPOLDINO
ORIENTADOR: SERGIO LUIS SARDI MERGEN
Local da Defesa e Data: Santa Maria, 16 de Dezembro de 2016.

Atas estão presentes na realidade de todos que frequentam reuniões. Elas são uma ferramenta de fácil confecção e de suma importância para o registro de informações discutidas em reuniões, e por serem assim são usadas em diversas áreas. Tipicamente uma ata é apresentada em uma ordem e escrita de forma corrida, na ordem que os temas são discutidos na reunião, sem preocupação com apresentação da informação, apenas seguindo a estrutura mencionada e linguagem correta. Essa forma de produzir uma ata é o que a torna de fácil execução, porém em termos de apresentação da informação, ela não é ideal. Para o leitor achar algum tópico de seu interesse, é necessário percorrer a ata inteira, fazendo uma leitura corrida do texto. E caso não se saiba em qual ata procurar, essa tarefa fica ainda mais onerosa. Uma forma de melhorar a apresentação da informações de uma ata seria resumi-la e estrutura-la em tópicos. Como assuntos diferentes são abordados numa ata, a busca por assuntos específicos é facilitada se a ata for organizada em tópicos hierárquicos, destacando os principais assuntos abordados e, a partir destes, estruturando subtópicos e sumarizando o seu conteúdo. Nesse contexto, este trabalho apresenta o desenvolvimento de algoritmos que sumarizam e estruturam em tópicos e subtópicos as atas de reuniões do colegiado do curso de Ciência da Computação da UFSM. Além disso, o trabalho apresenta um sistema que utiliza os algoritmos propostos para facilitar a navegação e busca por informações.

Palavras-chave: Hierarquização. Sumarização. Ata.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Atas estão presentes na realidade de todos que frequentam reuniões. Por definição (DICIO, 2016) ela é um registro escrito que contém os fatos, os acontecimentos e as resoluções de uma sessão, de uma assembleia, de uma convenção ou de uma reunião administrativa. Ela é uma ferramenta de fácil confecção e de suma importância para o registro de informações discutidas em reuniões, e por ser assim é usada em diversas áreas.

Tipicamente uma ata é apresentada na seguinte ordem: data da reunião, local, nome do presidente da entidade, ordem do dia, integrantes presentes, apreciação da ata, encerramento e assinaturas. Além dessa estrutura típica, uma ata é escrita de forma corrida, na ordem que os temas são discutidos na reunião, sem preocupação com apresentação da informação, apenas seguindo a estrutura mencionada e linguagem correta.

Essa forma de produzir uma ata é o que a torna de fácil execução, porém em termos de apresentação da informação, ela não é ideal. Para o leitor achar algum tópico de seu interesse, é necessário percorrer a ata inteira, fazendo uma leitura corrida do texto. Além disso, caso o leitor não tenha uma data de reunião específica em mente, apenas um tópico, essa tarefa fica mais onerosa, pois o mesmo terá que procurar em diversas atas pelo tópico de interesse.

O colegiado de Ciência da Computação da Universidade Federal de Santa Maria se reúne regularmente para discutir questões do curso, requisições de alunos, temas da faculdade no geral, etc. Como forma de registro, cada reunião é transcrita em uma ata. O relator da reunião segue a ordem e formas de escrita mencionadas anteriormente, e assim tem-se o mesmo resultado de outras atas no geral: fácil escrita, porém difícil pesquisa e leitura.

Uma forma de melhorar a apresentação da informações de uma ata seria resumi-la. Segundo Santos(2012), um resumo pode ser resultado da síntese de um ou mais documentos. Um resumo deve preservar a informação relevante e deve ser de tamanho reduzido. Numa área mais abrangente, podemos classificar um resumo como uma sumarização, que, segundo Rino (2003), tem como principal foco a tarefa de escrita de um texto condensado, o sumário, com o objetivo de transmitir ou comunicar somente o que é importante de uma fonte textual de informação. Em termos computacionais, de acordo com Santos(2012), a Sumarização Automática de Texto é o campo do Processamento de Linguagem Natural (PLN) que tem por objetivo produzir computacionalmente uma versão abreviada de um dado texto original mantendo o seu teor informativo.

Como assuntos diferentes são abordados numa ata, uma outra maneira de melhorar a apresentação da informação seria organizá-la em tópicos hierárquicos, destacando os principais assuntos abordados e, a partir destes, estruturando subtópicos. Nesse contexto, uma forma otimizada de apresentar informações de atas seria combinar sumarização automática com estruturação em tópicos hierárquicos.

Dentro desse escopo, este trabalho de graduação tem como objetivo desenvolver algoritmos que sumarizem e estruturem em tópicos e subtópicos as atas de reuniões do colegiado do curso de Ciência da Computação da UFSM, com o intuito de produzir estruturas hierárquicas de tópicos, com fácil apresentação para o usuário. Também tem-se o objetivo de desenvolver um sistema desktop para servir como instrumento de apresentação e navegação das informações para o usuário.

Este trabalho está estruturado da seguinte forma. O capítulo 2 apresenta a revisão bibliográfica, com conceitos usados no desenvolvimento do trabalho e como referência do estado da arte. O capítulo 3 mostra o desenvolvimento da solução através das seções 3.1, 3.2 e 3.3, respectivamente, a investigação dos padrões encontrado nas atas, a arquitetura das técnicas propostas, e a implementação dos algoritmos e interface visual. O capítulo 4 apresenta os resultados alcançados, onde na seção 4.1 é mostrada a aplicação desktop desenvolvida, e na seção 4.2 os experimentos realizados para averiguação da eficiência dos algoritmos. Por fim, o capítulo 5 apresenta as conclusões obtidas neste trabalho.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta alguns conceitos pesquisados neste trabalho, sendo usados diretamente na aplicação do mesmo ou apenas como referência do estado da arte.

2.1 Taxa de compressão

Santos (2012) define que a taxa de compressão é a medida indicadora da proporção entre o tamanho de um texto original($tam_{sumarizado}$) e o tamanho do resumo correspondente ($tam_{original}$), conforme indica a Equação 2.1:

$$taxa = \frac{tam_{sumarizado}}{tam_{original}} \tag{2.1}$$

Para exemplificar essa taxa, a Figura 2.1 mostra um texto original e a versão sumarizada do mesmo. O texto original possui 184 palavras, a versão sumarizada possui 36 palavras. Portanto a taxa de compressão em palavras é igual ao número de palavras do texto sumarizado dividido pelo número de palavras do texto original, resultando em 19,5% de taxa de compressão. O mesmo cálculo pode ser feito tomando como base o número de caracteres de cada texto.

2.2 Comparação de caracteres

Mergen et al (2005) propõe uma técnica de comparação de cadeias de caracteres, chamada Carla. Essa técnica ataca o problema de comparação entre cadeias onde, a maior diferença entre elas, é a ordem dos termos, e os termos não são separados por espaços em branco. A técnica utiliza a seguinte métrica para obter um valor de similaridade:

$$sim = \frac{|C|}{S_1 \times \alpha + S_2 \times \beta} \tag{2.2}$$

Sendo α + β = 1.0. As constantes α e β tem como objetivo dar um fator de importância para cada uma das cadeias comparadas. Essa métrica é considerada flexível pois se comporta de maneiras diferentes conforme a configuração usada nessas constantes.

Mergen et al (2005) explica que para o casamento de duas cadeias S_1 e S_2, é utilizada uma matriz nxm, sendo n igual ao numero de caracteres de S_1 e m igual ao numero de caracteres de S_2.

Texto Original:

Governo aprova 344 milhões para apoiar 89 mil jovens[1].
O Governo deverá aprovar na quarta-feira um pacote de medidas no valor de 344 milhões de euros para apoiar 89 mil jovens desempregados, disse à Lusa a secretária-geral da Confederação do Comércio e Serviços de Portugal (CCP).
A informação foi transmitida esta tarde às confederações patronais pelo ministro Adjunto e dos Assuntos Parlamentares, Miguel Relvas, no âmbito do programa "Impulso Jovem" e que será aprovado na quarta-feira em Conselho de Ministros, revelou Ana Vieira.
O montante, proveniente da reprogramação dos fundos comunitários Fundo Social Europeu e Fundo Europeu de Desenvolvimento Regional, destina-se a promover o emprego entre os jovens, cuja taxa de desemprego se situa nos 36,6 por cento.
Ana Vieira referiu ainda que, segundo Miguel Relvas, "o programa irá incidir sobre três eixos principais: estágios profissionais, apoios à contratação e apoios às empresas".
"Estas medidas serão um amortecedor e, globalmente, parecem-nos positivas", considerou a representante da CCP.
O ministro Miguel Relvas reuniu-se esta tarde com as confederações patronais para lhes dar conta da medida "Impulso Jovem", um dia antes desta ser aprovada pelo Governo.

1 – Diários de Notícias 05/06/2012

Sumário:

O Governo deverá aprovar na quarta-feira um pacote de medidas no valor de 344 milhões de euros para apoiar 89 mil jovens desempregados, disse à Lusa a secretária-geral da Confederação do Comércio e Serviços de Portugal (CCP).

Figura 2.1 – Texto original e sumarizado

Primeiramente, as posições da matriz $pos[x, y]$ sao preenchidas com uma informação que indica se o caractere x da cadeia S_1 é o mesmo do caractere y da cadeia S_2. Caso sejam diferentes, o valor é zero ($pos[x, y] = 0$). Caso contrário, o valor equivale a soma do valor da diagonal superior esquerda mais 1 (*pos[x, y] = pos*[x - 1, y - 1] + 1). A Figura 2.2 mostra exemplos de matrizes preenchidas dessa forma.

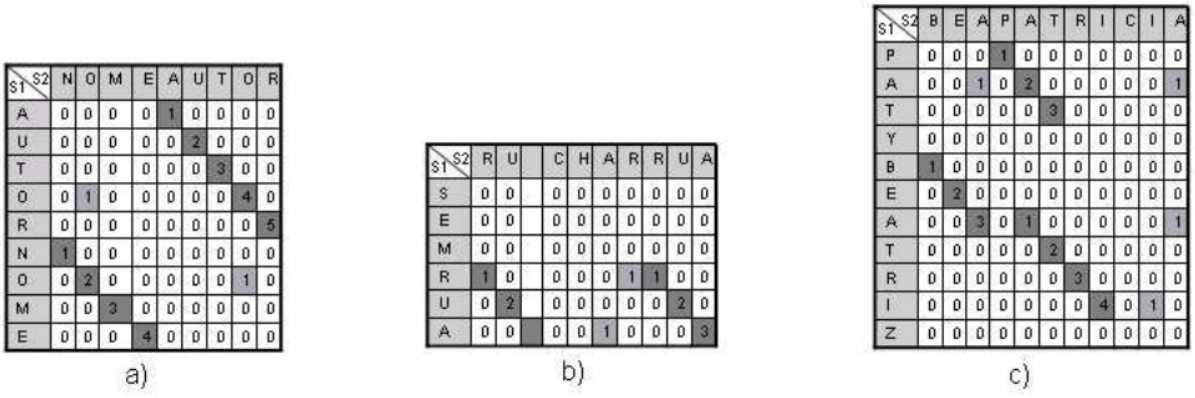

Figura 2.2 – Matrizes de comparação de cadeias

As células com tom de cinza mais escuro representam os termos em comum e que pos-

suem um mínimo de caracteres para formar um termo (no caso da Figura 2.2 esse mínimo é 3). As células com tom de cinza mais claro representam os termos incompletos (menos de 3 caracteres). A partir da matriz é possível obter o valor de C, que é a soma de todos os caracteres presentes nos termos completos. E então é possível calcular com a métrica o valor de similaridade.

2.3 Sumarização de textos

Conforme Santos (2012) cita, sumarização automática de texto é uma técnica para tratar a sumarização de forma automática, onde uma máquina resume e produz uma versão condensada do texto original, segundo determinados requisitos. Em geral, os sistemas de sumarização automática são tradicionalmente descritos como seguindo um modelo tripartido, ou seja, que possuem 3 fases, são elas: interpretação, transformação e geração.

Nesse modelo, a fase de interpretação corresponde à analise e interpretação do(s) textos(s) original(ais), para construir uma representação computável. Na fase transformação a representação obtida na etapa anterior é trabalhada no sentido de formular uma representação daquilo que deverá ser o resumo, ainda num formato computável. A fase geração trata da tradução da estrutura representativa do resumo para seu formato textual final.

Esse modelo é genérico o suficiente para definir os contornos gerais que devem definir um sistema de sumarização automática, porém, o ato de sumarizar é uma tarefa extremamente complexa.

No que diz respeito à metodologia de geração do sumário, um sistema de sumarização automática de texto poderá produzir um sumário extrativo ou abstrativo. O sumário extrativo é composto através da seleção e cópia de unidades textuais retiradas de forma literal do texto original, concatenando-as para compor o sumário. No sumário abstrativo o texto resumido é uma interpretação do texto original, o processo de produção envolve reescrever o texto original numa versão mais curta, substituindo conceitos extensos por outros análogos mas mais curtos.

Nas próximas sessões deste capítulo, são apresentados alguns métodos e trabalhos que usaram abordagens de sumarização abstrativas ou extrativas.

2.3.1 Métodos Clássicos

Santos (2012) cita três métodos como sendo clássicos da área de sumarização. O primeiro sugere que a frequência de uma determinada palavra num texto fornece uma medida útil da sua importância no texto. Numa fase inicial as palavras são reduzidas à sua forma canônica e palavras insignificantes são removidas. No passo seguinte as palavras significativas são ordenadas numa lista, por ordem descrescente das suas frequências, passando o índice de cada palavra na lista a representar o seu valor de significância. A importância de cada frase é depois calculada em função do número de palavras significativas entre cada ocorrência de uma palavra significativa. Na fase final as frases são ordenadas por ordem descrescente das suas importâncias, e as frases melhor classificadas são escolhidas para formar o sumário.

O segundo método defende que a informação saliente pode ser encontrada quando considerada a sua posição no documento. Santos (2012) cita que o autor do método analisou um conjunto de 200 parágrafos e concluiu que em 85% dos casos a frase mais representativa do tópico era a primeira e em 7% dos casos a última. Assim, no sentido de preservar a coesão textual, é sugerido que tanto a primeira como a última frase de cada parágrafo deveriam ser incluídas no sumário.

O terceiro método integra as características dos métodos anteriores e duas novas características, a presença de palavras sinalizadoras e a estrutura do documento. A pontuação final de uma frase é dada pela equação:

$$F_i = S_i \times p_1 + E_i \times p_2 + T_i \times p_3 + L_i \times p_4 \tag{2.3}$$

Onde F_i é a pontuação final da fase $_i$, e S_i, E_i, T_i e L_i são, respectivamente, a pontuação da frase $_i$ em função das palavras sinalizadoras que esta contém, das palavras estatisticamente relevantes que esta contém, das palavras presentes em elementos estruturalmente relevantes (títulos, cabeçalhos, etc.) que esta contém e da sua posição. Os pesos p_1 a p_4 definem a distribuição linear de importância atribuída às quatro pontuações parciais.

2.3.2 Método da Ideia Principal

Pardo (2002) apresenta um sumarizador automático de textos chamado GistSumm (GIST SUMMarizer), o qual procura simular a forma de sumarizar humana, que é procurando pela ideia principal do texto e as informações que a complementam.

O GistSumm tem como cerne um ranqueamento de sentenças, que é executado por um método de ranqueamento selecionado pelo usuário, que são: palavras-chave ou TF-ISF (*Term Frequency-Inverse Sentence Frequency*). A sentença melhor ranqueada é considerada a ideia principal, a partir da qual as outras sentenças que comporão o sumário serão determinadas.

Detalhadamente, o processo de ranqueamento ocorre após as sentenças terem sido separadas por um segmentador sequencial. Depois disso são criados vetores das sentenças, onde cada palavra é colocada em um posição diferente, e então três técnicas são aplicadas para aprimorar os resultados da sumarização, são eles: *case folding* (transforma todas as letras em minúsculas), *stemming* (determina a forma canônica de uma palavra) e remoção de *stopwords* (remove palavas consideradas irrelevantes).

Após isso, a pontuação das sentenças é efetuada pelo uso do método das palavras-chave ou TF-ISF. Pelo primeiro método, cada vetor recebe como pontuação a soma do número de ocorrências de cada uma de suas palavras no texto inteiro, ou seja, em todos os vetores.

Pelo segundo método, a pontuação do vetor corresponde à média da pontuação de cada uma de suas palavras. A pontuação de cada palavra, por sua vez, é calculada da seguinte forma:

$$P = w_1 \times log((t)/(w_2)) \tag{2.4}$$

Sendo P a pontuação de uma palavra W, w_1 o número de vezes que a palavra W ocorreu na sentença, t o número total de palavras da sentença, e w_2 o número de sentenças em que a palavra W ocorreu.

Por ambos os métodos, a sentença com maior pontuação é considerada como sendo a ideia principal do texto. A partir dessa sentença, o GistSumm seleciona as sentenças que formarão o sumário, e para isso, 2 passos são executados.

Primeiramente é calculada a média da pontuação das sentenças do texto-fonte e assume essa média como sendo a *baseline* para corte das possíveis sentenças que formarão o sumário. Após, são selecionadas, juntamente com a sentença que representa a ideia principal, todas as sentenças que: contenham pelo menos uma palavra entre as palavras da ideia principal, ou, possuam uma pontuação maior que a *baseline* calculada no primeiro passo.

2.3.3 Método Baseado em Lógica *Fuzzy*

Goularte (2015) apresenta o FSUMM, um sumarizador automático que tem como núcleo a utilização de lógica *fuzzy* para produzir sumários, com base nas características estatísticas e

linguísticas do texto. O FSUMM é dividido em três processos principais: pré-processamento, análise das características textuais e análise *fuzzy*.

No pré-processamento são realizadas a *tokenização*, a remoção de *stop words* e a segmentação do texto. No segundo processo, é feito o cálculo estatístico das características, sendo elas: posição, comprimento, título e palavras-chave. Essas características foram citadas nos métodos explorados nos tópicos anteriores desta revisão.

No terceiro processo, a análise *Fuzzy*, analisadores difusos são usados para classificar as sentenças com base nas estatísticas da etapa anterior. Nessa etapa, o autor segue seis passos para calcular a saída. São elas:

- Determinar o conjunto de regras *fuzzy*;
- *Fuzzyficar* as entradas usando funções de pertinências de entrada;
- Combinar as entradas *fuzzyficadas* de acordo com as regras *fuzzy* para estabelecer a força da regra;
- Encontrar o consequente da regra para combinar a regra forte com a função de pertinência de saída;
- Combinar os consequentes para obter uma distribuição de saída;
- *Desfuzzyficar* a distribuição de saída.

O autor determina três variáveis de entrada, sendo elas: posição e comprimento (loc-len) e palavras-chave (K_1 e K_2). Essas são as saídas do anterior, de análise das características. Essas entradas são utilizadas em um sistema de regras *fuzzy*, onde dependendo do valor que as variáveis assumem, em um conjunto previamente especificado (Baixo, Médio ou Alto), é gerada uma saída, denominada informativade, a qual também assume um valor daquele conjunto.

O autor definiu 27 regras, as quais seguem a forma do exemplo a seguir:

R_1: Se (loc-len é Baixo) e (K_1 é Baixo) e (K_2 é Baixo) Então (Informatividade é Baixa)

As variáveis tem seus valores calculados através de funções de pertinência, nesse caso do tipo gaussianas, e com esses valores são geradas saídas *desfuzzyficadas*, ou seja, valores reais que então são usados como métricas de avaliação, para definir se a sentença é informativa o suficiente para constar no sumário. Assim, ao fim do processo de análise *fuzzy*, é gerado o sumário com as sentenças de maior valor de informativadade.

3 DESENVOLVIMENTO DA SOLUÇÃO

Este capítulo apresenta a modelagem do problema e sua implementação. Aqui são apresentados exemplos de atas do colegiado do curso de ciência da computação, padrões encontrados nessas atas, a modelagem dos algoritmos derivados a partir destes padrões e suas implementações.

3.1 Investigação dos padrões encontrados

Um total de 76 atas foram usadas como base de estudo e experimentação neste trabalho. Em um primeiro momento elas foram investigadas a fim de encontrar padrões que pudessem ser usados como base lógica de estruturação. As seções a seguir apresentam o padrão de ordenação básico das atas e diferentes padrões de estruturação encontrados.

3.1.1 Padrão de ordenação

Como é descrito na introdução, as atas costumam seguir uma ordem e estrutura, com algumas variações. A Figura 3.1 mostra um exemplo de ata do colegiado de Ciência da Computação.

Na Figura é mostrada a parte inicial da ata onde é possível notar a seguinte ordenação: data da reunião, horário, local, presidência, ordem do dia, lista de pessoas presentes, discussão dos itens da ordem do dia. Essa ordenação foi percebida na maioria das atas do colegiado, e assim foi definida como o padrão de ordenação das atas para esse trabalho. Casos onde a ordenação aparece diferente são tratados como exceção.

3.1.2 Padrões de estruturação

Além do padrão de ordenação definido, há padrões de estruturação nas atas, isto é, as formas como são mostradas a ordem do dia, as pontuações usadas para delimitar o tema de uma ordem do dia, palavras e pontuações que definem o final da lista de ordem do dia, entre outros.

O primeiro padrão é o da lista de ordens do dia, que aqui serão tratados como os tópicos principais de uma ata. As Figuras 3.2 e 3.3 exemplificam o padrão na listagem das ordens do dia. Nelas é possível notar que cada tópico é definido com a sequência mostrada na Figura 3.4: um número, um sinal de ponto, o nome do tópico, um sinal de ponto-e-vírgula. Também é

Ata nº 115/2014

Aos vinte e oito dias do mês de novembro do ano de dois mil e quatorze, às nove horas, na sala 321-A do Centro de Tecnologia, sob a presidência da professora Andrea Schwertner Charão, coordenadora do Curso de Ciência da Computação, reuniu-se o colegiado do curso de Ciência da Computação, em uma sessão ordinária, com a seguinte ordem do dia: **1.Apreciação da ata n° 114; 2.Apreciação de ACG's; 3.Apreciação de PIEC's; 4.Homologação de resultado de seleção de ingresso/reingresso; 5.Apresentação de resultados de avaliação do curso pelos alunos; 6.Outros assuntos; 7.Comunicações.** Presentes os professores Andrea Schwertner Charão, Patricia Pitthan de Araújo Barcelos, Benhur de Oliveira Stein, Luis Alvaro de Lima Silva e Iara Augustin. A presidente abriu a sessão e em seguida questionou se haveria alguma sugestão de alteração de pauta. Não havendo novos itens, passou-se então à ordem do dia. **1.Apreciação da ata nº 114.** A ata nº 114 foi disponibilizada on-line aos membros do colegiado. A presidente colocou a ata em aprovação. **Deliberação:** Aprovada. **2.Apreciação de ACG's.** O acadêmico **Alberto Francisco Kummer Neto** solicitou a seguintes ACG's: Publicação de trabalho (internacional). **Deliberação**: Anexar comprovante da publicação e justificativa relativa à entrega da solicitação fora de prazo e retornar para avaliação. Publicação de trabalho (regional). **Deliberação:** Aprovado 4 horas. Estágio extracurricular. **Deliberação:** Aprovado 540 horas. Participação em eventos. **Deliberação:** Aprovado 38 horas. O acadêmico **Alexandre Cervi Soares** solicitou as seguintes ACG's: Atividade de extensão (Clube de Computação). **Deliberação:** Favorável a 30 horas mediante apresentação de registro no Gabinete de Projetos da UFSM. Atividade de extensão (Descubra UFSM). **Deliberação:** Aprovado 36 horas. Estágio extracurricular. **Deliberação:** Aprovado 664 horas. Participação em eventos. **Deliberação:** Aprovado 57 horas. Monitoria. **Deliberação:** Aprovado 45 horas. O acadêmico **Tiago Boelter Mizdal** solicitou as seguintes ACG's: Monitoria. **Deliberação:** Aprovado

Figura 3.1 – Exemplo de ata do colegiado do curso de Ciência da Computação da UFSM

possível perceber que a listagem dos tópicos acaba quando a listagem das pessoas presentes à reunião começa, ou seja, acaba antes da palavra "Presentes".

Computação, em uma sessão ordinária, com a seguinte ordem do dia: **1.Apreciação da ata n° 114; 2.Apreciação de ACG's; 3.Apreciação de PIEC's; 4.Homologação de resultado de seleção de ingresso/reingresso; 5.Apresentação de resultados de avaliação do curso pelos alunos; 6.Outros assuntos; 7.Comunicações.** Presentes os professores Andrea Schwertner Charão, Patricia

Figura 3.2 – Exemplo de estrutura de ordens do dia em uma ata do colegiado do Curso de Ciência da Computação da UFSM

sessão ordinária, com a seguinte ordem do dia: **1.Apreciação da ata n° 109 (relator Prof Luis Alvaro Lima Silva); 2.Apreciação de ACG's; 3.Apreciação de PIEC; 4.Vagas para intercambista Convênio BRACOL; 5.Recepção de aluno oriundo da Ostfália University; 6.Edital CRIA-I 2014; 7. Outros assuntos; 8.Comunicações.** Presentes os professores Andrea Schwertner Charão, Patricia Pitthan de

Figura 3.3 – Exemplo de estrutura de ordens do dia em uma ata do colegiado do Curso de Ciência da Computação da UFSM

O segundo padrão importante neste trabalho é o referente aos conteúdo da ordem do dia, ou seja o conteúdo de um tópico. As Figuras 3.5 e 3.6 mostram alguns exemplos e referem-se as mesmas atas das Figuras 3.2 e 3.3, respectivamente. Nota-se que o conteúdo de um tópico é descrito logo após seu nome, que é o mesmo que consta na lista de ordens do dia mencionado anteriormente, e se não esta escrito de forma igual, é muito similar.

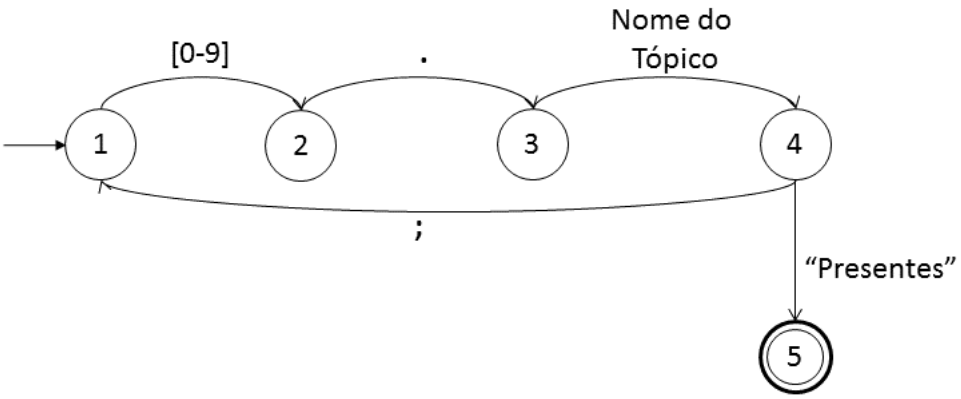

Figura 3.4 – Diagrama de eventos da listagem da ordem do dia de uma ata do colegiado do Curso de Ciência da Computação da UFSM

> Não havendo novos itens, passou-se então à ordem do dia. **1.Apreciação da ata nº 114.** A ata nº 114 foi disponibilizada on-line aos membros do colegiado. A presidente colocou a ata em aprovação. **Deliberação:** Aprovada. **2.Apreciação de ACG's.** O acadêmico **Alberto Francisco Kummer Neto**

Figura 3.5 – Exemplo de estrutura do conteúdo de um tópico de uma ata do colegiado do Curso de Ciência da Computação da UFSM

> plano do acadêmico. **4.Vagas para intercambista Convênio BRACOL.** Em resposta à consulta encaminhada à Coordenação do Curso de Ciência da Computação acerca de vagas para o convênio BRACOL, o Colegiado decidiu oferecer duas vagas para o primeiro semestre letivo de 2015. **5.Recepção**

Figura 3.6 – Exemplo de estrutura do conteúdo de um tópico de uma ata do colegiado do Curso de Ciência da Computação da UFSM

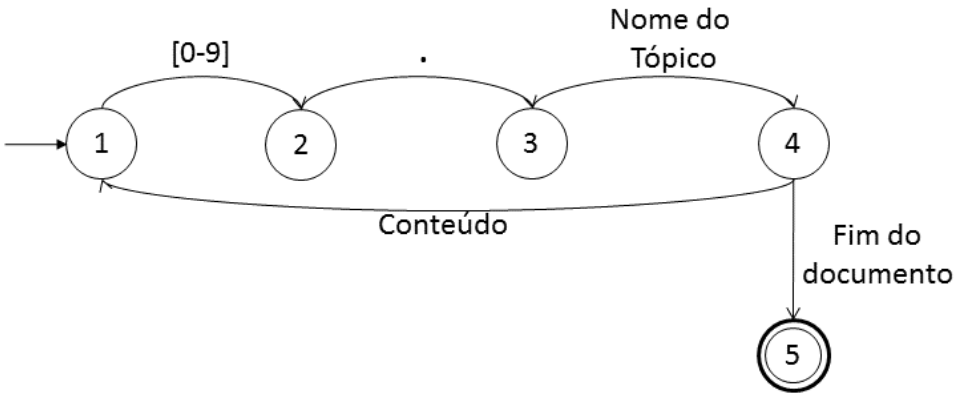

Figura 3.7 – Diagrama de eventos da estrutura do conteúdo de um tópico de uma ata do colegiado do Curso de Ciência da Computação da UFSM

Portanto o padrão nesse caso segue o formato mostrado na Figura 3.7: o conteúdo começa depois do nome do tópico, ou seja, depois da sequência de número, ponto, nome do tópico, ponto; O conteúdo acaba antes do ínicio de outro conteúdo, ou seja, antes do nome do próximo tópico; No caso do último tópico, o fim do conteúdo é o fim do documento.

O terceiro padrão refere-se a tópicos específicos de apreciação de solicitações de ACG's

e PIEC's. Nestes tópicos existem repetições de alguns termos que podem serem entendidos como subtópicos.

A Figura 3.8 mostra que, no caso de uma solicitação de ACG, é descrito na ata o nome do aluno solicitante, nome da ACG, deliberação (aprovado ou reprovado), número de horas. Como existem muitas solicitações de ACG's, pode-se inferir como subtópicos as solicitações de cada aluno. O padrão encontrado é o mostrado na Figura 3.9: palavra "acad", ponto, nome do aluno, frase "solicitou as seguintes ACGs", dois pontos, nome da ACG, ponto, palavra "deliberação", dois pontos, palavra "aprovado"ou "reprovado", número de horas, palavra "horas", ponto; Depois é descrita outra ACG requisitada ou o início do pedido de outro aluno, ou seja, a palavra "acad".

Científica. **Deliberação:** Aprovado 720 horas. Monitoria. **Deliberação:** Aprovado 10 horas. O acad. **Luiz José Schirmer Silva** solicitou as seguintes ACGs: Atividades de Iniciação Científica. **Deliberação:** Aprovado 480 horas. Monitoria. **Deliberação:** Aprovado 30 horas. O acad. **Emmanuel Katende Dinanga** solicitou as seguintes ACGs: Participação em Eventos. **Deliberação:** Aprovado 7 horas. Atuação em Núcleo Temático. **Deliberação:** Aprovado 226 horas. Publicação de trabalhos. **Deliberação:** Aprovado 16 horas. O acad. **Vitor da Silva** solicitou as seguintes ACGs: Participação em Eventos. **Deliberação:** Aprovado 44

Figura 3.8 – Exemplo de subtópicos de ACGs de uma ata do colegiado do Curso de Ciência da Computação da UFSM

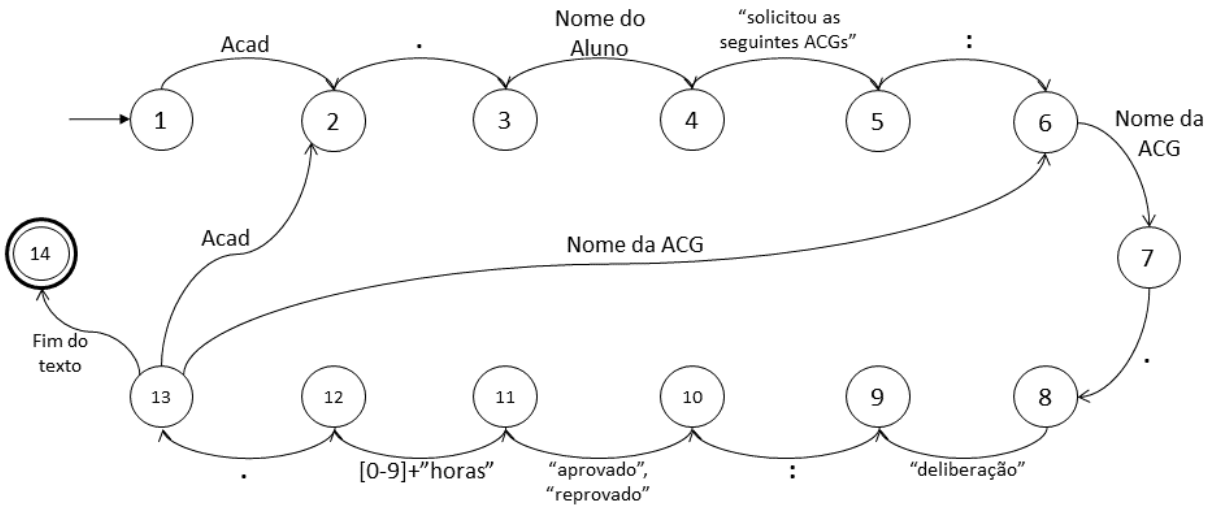

Figura 3.9 – Diagrama de eventos de subtópicos de ACGs de uma ata do colegiado do Curso de Ciência da Computação da UFSM

O quarto padrão é mostrado na Figura 3.10, nela estão subtópicos do tipo número mais letra, ou seja, subtópicos definidos explicitamente pelo relator da ata. Neste caso o padrão é simples, como mostrado na Figura 3.11: número do tópico mais letra, pontuação (ponto, dois pontos, ponto-e-vírgula, etc), conteúdo do subtópico; Depois começa outro subtópico, com a

próxima letra do alfabeto, ou começa outro tópico.

item n° 6 da mesma, o que foi aceito por todos os conselheiros. Passando para as comunicações: **1.Comunicações. 1a.** Prof. Raul informou que o DELC recebeu duas vagas para docentes, sendo uma para a área de eletrônica e outra para a área de computação. O concurso para computação será para professor adjunto na área de Fundamentos para Computação. **1b**. Prof. Raul comunicou que estará recebendo as sugestões de aquisição dos livros para a biblioteca central até às doze horas do dia 06/05/2005, quando então encaminhará a lista. **1c**. Prof. Raul recebeu através do gabinete do reitor um memorando comunicando que foram doados quatro exemplares de livros e os mesmos encontram-se a disposição na biblioteca setorial do Centro de Tecnologia. **1d**. Prof. Raul comunicou que encerraram as inscrições para o curso de especialização em Sistemas de Computação para WEB o qual teve 23 inscritos pagantes e 2 não pagantes do quadro da UFSM. **1e**. Prof. Raul informou que a resolução n. 005/2005 altera a denominação do Núcleo de Propriedade Intelectual – NPI – da PRPGP para Núcleo de Inovação e Transferência de Tecnologia–NIT– e expede nova regulamentação, revogando a Resolução n.010/01. **1f.** Prof. Raul comunicou que recebeu um memorando da

Figura 3.10 – Exemplo de subtópicos do tipo número mais letra de uma ata do colegiado do Curso de Ciência da Computação da UFSM

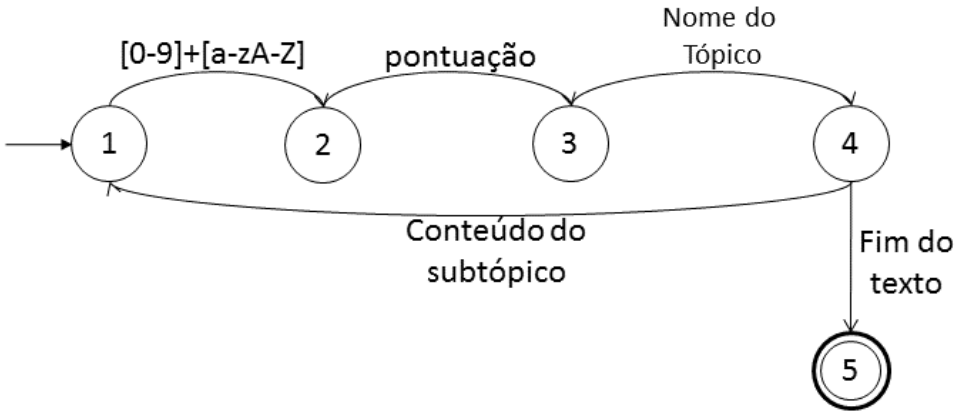

Figura 3.11 – Diagrama de eventos de subtópicos do tipo número mais letra de uma ata do colegiado do Curso de Ciência da Computação da UFSM

3.2 Arquitetura proposta

A partir dos padrões de estruturação encontrados nas atas, foi elaborada a arquitetura que engloba técnicas usadas para a extração das informações e hierarquização. O algoritmo foi organizado em módulos, como mostra a Figura 3.12.

Os módulos referem-se as tarefas mais importantes no algoritmo, são elas: identificação dos tópicos; Delimitação do conteúdo de um tópico; Subdivisão do conteúdo em subtópicos; Sumarização do conteúdo. O algoritmo tem como entrada o texto original da ata, e ao final tem-se como saída a estrutura em tópicos. A seguir são detalhados os algoritmos de cada módulo.

3.2.1 Técnica auxiliar

A técnica a seguir é usada nos algoritmos principais.

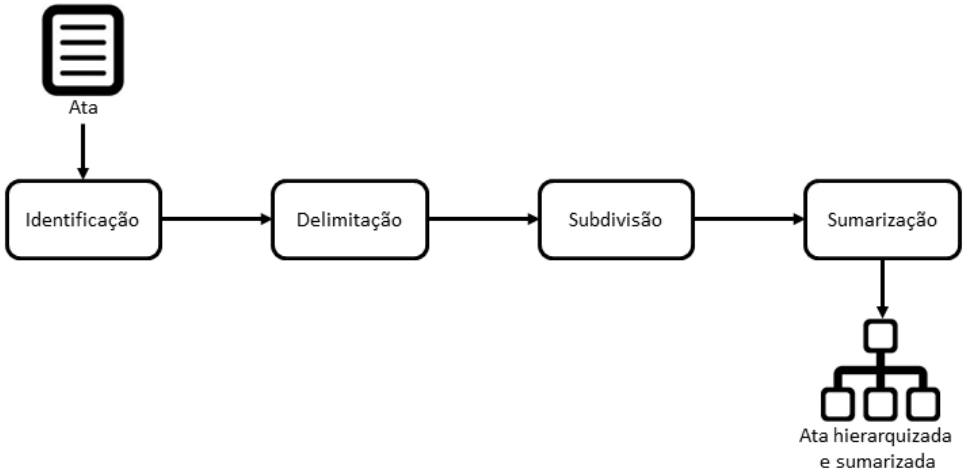

Figura 3.12 – Arquitetura da Solução

Algoritmo 1: buscaTexto

Entrada: Vetores de strings: $vet1$, $vet2$, Inteiro: $posInicial$
Saída: $pos1$
início
  Inteiro $pos1$=-1, $pos2$=FINAL_TEXTO,aux=-1;
  para *cada* $i \in vet2$ **faça**
    aux = indexOf(i,$posInicial$);
    if aux*!=-1 &&* aux<$pos2$ **then**
      $pos2$ = aux;
  fim
  aux = -1;
  para *cada* $i \in vet1$ **faça**
    aux = indexOf(i,$posInicial$);
    if aux*!=-1 &&* aux<$pos2$ **then**
      $pos1$ = aux;
  fim
  retorna $pos1$;
fim

Essa técnica tem como objetivo retornar a posição de uma *string* no texto, para isso ela recebe dois vetores de *strings*, o primeiro referente as *strings* que espera-se encontrar, o segundo referente a palavras que definem um limite na busca, e também recebe um inteiro que indica a posição inicial da busca. Primeiramente é encontrada a posição da palavra limite mais próxima, após isso um laço de repetição é efetuado, onde é procurada a posição das *strings* do primeiro vetor, a posição de menor valor será a posição retornada, pois refere-se a primeira ocorrência de uma das *strings* no texto.

3.2.2 Identificação dos tópicos

O algoritmo 2 mostra o processo para extração da lista dos tópicos de uma ata.

Algoritmo 2: Identifica tópicos

```
Entrada: texto da ata
Saída: lista de tópicos
início
    Inteiro num=1, pos1, pos2=0;
    String s;
    Lista<String> topicos;
    repita
        pos1 = buscaTexto({concat(num,'.')},{'Presentes'},pos2);
        num = num + 1;
        pos2 = buscaTexto({concat(num,'.')},{'Presentes'},pos1);
        if pos1!=-1 && pos2!=-1 then
            s = substring(pos1 + tam(pos1), pos2);
            topicos = topicos U s;
    até pos1==-1 || pos2==-1;
fim
```

Um laço de repetição é efetuado e a cada iteração é procurado a posição de *num* mais um sinal de ponto, *num* sendo um inteiro que começa em 1, após *num* é incrementado em mais 1, e é procurada no texto a próxima posição de *num* mais um sinal de ponto. Caso a primeira e segunda posições sejam encontradas, o conteúdo entre elas é extraído do texto e adicionado a uma lista de tópicos.

Como comentado anteriormente a palavra "Presentes"identifica o fim da listagem de tópicos, assim as buscas vão até essa palavra, que é passada como paramêtro na função *buscaTexto*. O laço termina quando não forem mais encontradas posições válidas e após isso o algoritmo retorna a lista de tópicos.

3.2.3 Delimitação do conteúdo

O módulo de delimitação do conteúdo de um tópico é apresentado no algoritmo 3.

Algoritmo 3: Delimita conteúdo de um tópico

```
Entrada: texto da ata, tópico, posição inicial de busca
Saída: Posição de inicio do conteúdo
início
    Lista<String> possiveisTitulos;
    Inteiro pos, num, tam;
    Float score, melhorScore;
    String s, titulo;
    num = número do tópico;
    tam = tamanho do nome do tópico;
    repita
        pos = buscaTexto({concat(num,'.')},{},posInicial);
        if pos!=-1 then
            s = substring(pos,pos+tam);
            possiveisTitulos = possiveisTitulos U s;
            posInicial = pos;
    até pos!=-1;
    melhorScore = 0.0;
    para cada i ∈ possiveisTitulos faça
        score = similaridade(i,nome_topico);
        if score > melhorScore then
            titulo = i;
            melhorScore = score;
    fim
fim
```

A ideia desse algoritmo é procurar no texto, partindo da posição depois da listagem dos tópicos, os nomes dos tópicos no corpo do texto, considerando que eles indicam o início do conteúdo do mesmo. Sendo assim essa busca é feita primeiramente pelo número do tópico mais o sinal de ponto, porém como podem haver mais números com sinal de ponto, indicando algum valor fracionário por exemplo, o algoritmo busca todos os números mais ponto.

Para cada posição encontrada, é extraído do texto um possível nome de tópico, uma *string* entre a posição encontrada mais o tamanho do nome do tópico, e isso é salvo em uma lista. A seguir um novo laço de repetição percorre a lista de possíveis nomes de tópicos e compara com o nome real do tópico, o que obtiver a melhor porcentagem de similaridade é definido como o início do conteúdo do tópico. Essa comparação é feita com a técnica Carla, mencionada anteriormente.

Esse algoritmo é aplicado para todos os tópicos encontrados, assim obtem-se a posição inicial de todos os conteúdos. A posição final do conteúdo é a posição inicial do conteúdo seguinte, e no caso do último conteúdo, a posição final é o fim do documento.

3.2.4 Subdivisão em subtópicos

Após os módulos de identificação de tópicos e delimitação de conteúdo, o algoritmo possui uma lista de tópicos e seus conteúdos. Nessa fase é efetuada uma análise dos conteúdos e determina-se quais possuem subtópicos, quais necessitam de sumarização e quais não precisam de mudanças.

Foi analisado que na maioria das atas os conteúdos são pequenos, não possuindo subtópicos. Porém, alguns conteúdos específicos possuem conteúdo grande. São as Apreciações de ACGs e PIECs. Esses conteúdos são constituídos de requisições de alunos para que suas ACGs e PIECs sejam aprovados pelo colegiado, e em várias atas há muitas requisições. Portanto foi definido que o módulo de subdivisão em subtópicos é aplicado somente nesses tópicos e naqueles que os subtópicos são explicitamente especificados com subitens do tipo número mais letra.

3.2.4.1 Subdivisão dos tópicos de ACGs

Algoritmo 4: Subdivide tópicos de ACGs

```
Entrada: tópico
Saída: subsEstudantes
início
    Lista<String> requisicoes,deliberacoes;
    Lista<Topico> subsEstudantes, subsACGs;
    Topico t1, t2;
    Inteiro pos1=0, pos2=0, pos3, pos4, proxEst;
    String r, d;
    repita
        pos1 = buscaTexto({'acad','academico'},{},pos1);
        pos2 = buscaTexto({'solicitou'},{},pos2);
        if pos1!=-1 && pos2!=-1 && pos2-pos1<50 then
            estudante = substring(pos1,pos2);
            pos3 = buscaTexto({'acg:','acg.','acg '},{},pos2) + 'acg
            '.tamanho();
            pos4 = buscaTexto({'deliberação'},{},pos3) - 1;
            proxEst = buscaTexto({'acad'},{},pos4);
            repita
                r = substring(pos3,pos4);
                pos3 = buscaTexto({'deliberação'},{},pos4) +
                'deliberação'.tamanho();
                pos4 = buscaTexto({'.',':'},{},pos3);
                d = substring(pos3,pos4);
                subsACGs.adicionaNaListaTopicos(novoTopico(r,d));
                pos3 = pos4 + 1;
                pos4 = buscaTexto({'deliberação'},{},pos3);
            até pos4<proxEst && pos4>0;
            t2.adicionaListaSubtopicos(subsACGs);
            t2.adicionaTitulo(estudante);
            subsEstudantes U t2;
    até pos1!=-1 && pos2!=-1;
    retorna subsEstudantes;
fim
```

O algoritmo 4 detalha a subdivisão de tópicos para os tópicos de apreciações de ACGs. Nele o laço mais externo refere-se a busca da solicitação de cada aluno, até não encontrar mais alunos. Dentro do laço, *pos1* recebe a posição do próximo "acad"ou "academico"(a) no texto, *pos2* recebe a posição de "solicitou", entre *pos1* e *pos2* estará o nome de um aluno. Caso não ache nenhum, sai do laço. Caso o *pos2-pos1* seja maior que 50, também sai do laço, pois o *pos1* achado não foi referente a um aluno, mas sim a palavra "acadêmico"usada em outro contexto.

Caso passe no teste, *estudante* armazena o texto entre *pos1* e *pos2*. A variável *pos3* armazena a posição do próximo "acg:", "acg."ou "acg "e o tamanho dessa *string*. A posição do próximo "deliberação"menos 1 é armazenada em *pos4* e a posição do próximo "acad"ou variações é armazenada em *proxEst*.

Após isso, o laço interno é iniciado, nele a *string r* armazena o texto entre *pos3* e *pos4*, o qual refere-se a uma ACG requisitada. Depois *pos3* recebe a posição do próximo "deliberação"mais o tamanho dessa *string*, *pos4* recebe a posição do próximo sinal de ponto ou dois pontos. A *string d* armazena o texto entre *pos3* e *pos4*, referente a deliberação da ACG. Com a requisição *r* e a deliberação *d* é criado um novo tópico que é armazenado na lista de tópicos *subsACGs*.

Então *pos3* é atualizada para *pos4* mais 1, e *pos4* recebe a posição do próximo "deliberação". O laço acaba quando *pos4* não for mais encontrada ou for maior que *proxEst*, que define o inicio da requisição de outro estudante. Depois, a *string estudante* e a lista *subsACGs* são adicionadas como um novo tópico na lista de tópicos *subsEstudantes*. Após o término do laço mais externo, o algoritmo retorna a lista de tópicos *subsEstudantes*.

3.2.4.2 Subdivisão dos tópicos de PIECs

Algoritmo 5: Subdivide tópicos de PIECs

```
Entrada: tópico
Saída: subsEstudantes
início
    Lista<Topico> subsEstudantes, subsPIECs;
    Topico t1, t2;
    Inteiro pos1, pos2=0, pos3, pos4, proxEst;
    String r, d;
    repita
        pos1 = buscaTexto({'acad','acadêmico'},{},pos2);
        pos2 = buscaTexto({'solicitou'},{},pos1);
        if pos1!=-1 && pos2!=-1 && pos2-pos1<50 then
            estudante = substring(pos,pos2);
            pos3 = buscaTexto({'graduação'},{},pos2) +
            'graduação'.tamanho();
            pos4 = buscaTexto({'deliberação'},{},pos3) - 1;
            r = substring(pos3,pos4);
            proxEst = buscaTexto({'acad','acadêmico',},{},pos4);
            d = substring(pos4,proxEst);
            t1.adicionaTitulo(r);
            t1.adicionaConteudo(d);
            subsPIECs U t1;
            t2.setListaSubtopicos(subPIECs);
            t2.adicionaTitulo(estudante);
            subsEstudantes U t2;
    até pos1!=-1 && pos2!=-1;
    retorna subsEstudantes;
fim
```

O algoritmo 5 mostra o processo para extração dos subtópicos de um tópico de PIECs. Ele segue o mesmo modelo do algoritmo 4, porém não precisa do laço interno, já que um aluno sempre pede apenas um PIEC, assim um aluno tem um PIEC como subtópico e o PIEC tem apenas uma deliberação como subtópico.

3.2.4.3 Subdivisão dos tópicos do tipo número mais letra

Algoritmo 6: Subdivide tópicos do tipo número mais letra

```
Entrada: tópico
Saída: subTópicos
início
    Lista<Topico> subTopicos;
    Lista<Inteiro> subPosicoes;
    Topico t;
    Inteiro pos,pos1, pos2, indice;
    char ch;
    String item, s1, s2;
    pos1=pos2=0;
    ch = 'a';
    indice = indice do tópico;
    item = concat(indice,ch);
    repita
        pos = buscaTexto({concat(item,'.'),concat(item,':'),concat(item,'
        ')},{},pos);
        if pos!=-1 then
            subPosicoes U pos;
            ch++;
            item = concat(indice,ch);
    até pos!=-1;
    para cada i ∈ subPosicoes faça
        s1 = substring(i,i+2);
        if i==subPosicoes.tamanho() then
            s2 = substring(i+2,texto.tamanho());
        else
            s2 = substring(i+2,i.proximo());
        end
        t.adicionaTitulo(s1);
        t.adicionaConteudo(s2);
        subTopicos U t;
    fim
    retorna subTopicos;
fim
```

O algoritmo 6 trata dos casos em que um tópico possui divisão em subtópicos do tipo número mais letra, por exemplo: "1a.", "1b:", "1c ", etc. Nele há 2 laços, no primeiro são procurados as posições de cada "número mais letra mais pontuação"e cada posição é armazenada numa lista de posições.

Assim, *s1* refere-se ao título do subtópico, e extraí o texto entre a posição armazenada

em *i* e esse mesmo valor mais 2 posições, ou seja, o número mais letra. Já *s2* é o conteúdo do subtópico, que é extraído do texto entre a posição armazenada em *i* mais 2 posições e o valor de *i* na próxima iteração, ou, o tamanho do texto, caso *i* tenha atingido o limite da lista.

As *strings s1* e *s2* são armazendos em um tópico *t* e ele é adicionado na lista de tópicos *subTopicos*, o qual retorna ao fim do laço.

3.2.5 Sumarização do conteúdo

O módulo de sumarização é aplicado em conteúdos que não foram subdividos em subtópicos, e apenas naqueles que possuem mais que *X* palavras, *X* sendo pré-definido previamente. Na análise de modelagem deste módulo, foi notado que os conteúdos de tópicos da maioria das atas não costumam ser grandes, portanto não requerem uma sumarização complexa. Dessa forma foi definida como método de sumarização apenas a remoção de *stop-words*, ou seja, apenas a remoção de palavras armazenadas em uma lista e que não são relevantes na compreensão do texto.

3.3 Algoritmo e Interface Visual: Implementação

Esta seção detalha a implementação do algoritmo de hierarquização de uma ata, e também mostra o desenvolvimento do sistema *desktop* feita para a visualização e navegação dos tópicos. O desenvolvimento foi realizado com a linguagem de programação Java, sua escolha foi feita pela maior familiaridade do autor com a mesma.

3.3.1 Arquitetura

O sistema foi desenvolvido baseado na arquitetura MVC *(Model-View-Control)*, o qual separa a parte visual, a parte de dados e a parte de processamento, deixando assim o código melhor modularizado, e assim, mais adaptável. A Figura 3.13 mostra o diagrama de classes do sistema.

3.3.1.1 Classe: Topic

A classe *Topic* é utilizada para armazenar as informações de um tópico. Ela contém o *strings* referentes ao título e conteúdo do tópico, listas de *string* contendo o título e conteúdo *tokenizados*, um inteiro referente ao índice, e uma lista de *Topic* aonde são armazenados os

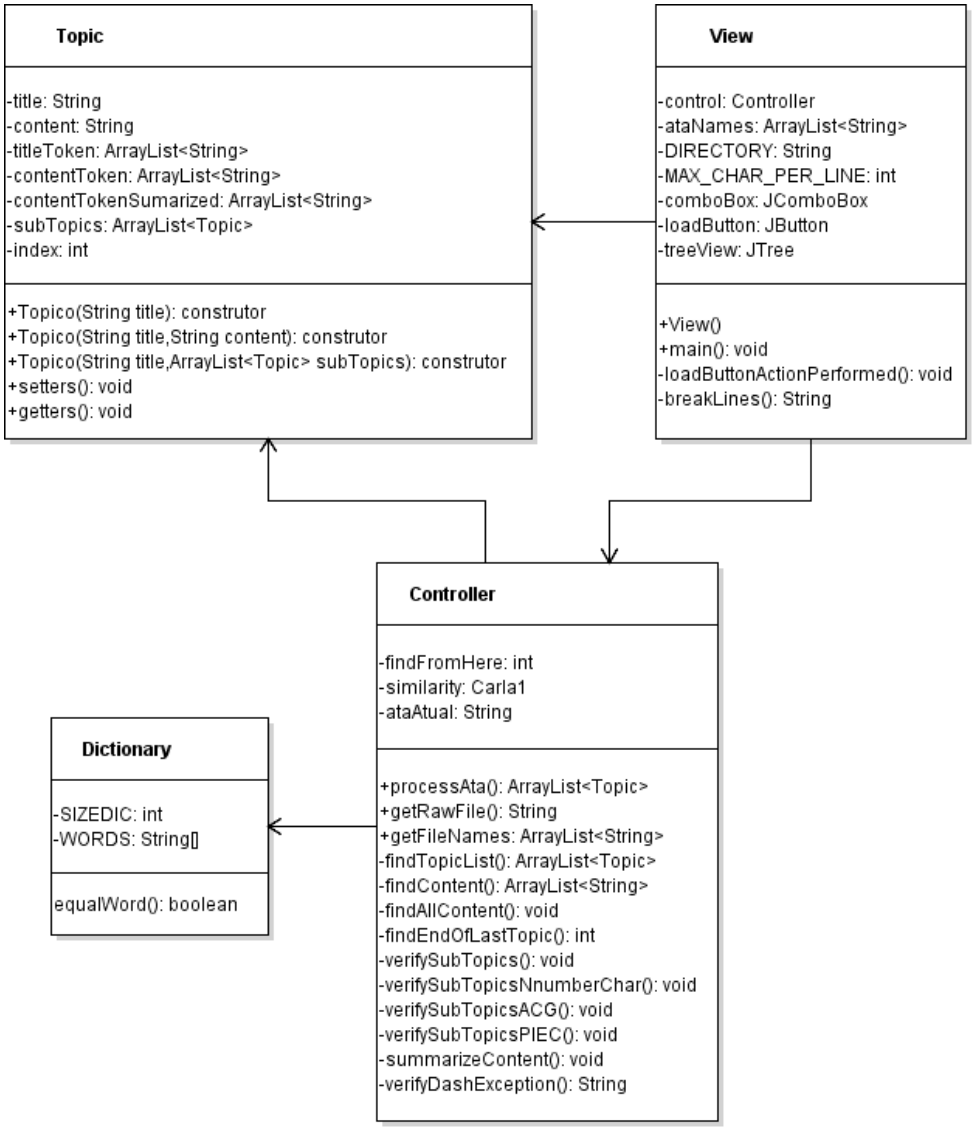

Figura 3.13 – Diagrama de classes do sistema desktop

subtópicos, caso esses existam. A classe *Dictionary* tem a finalidade de ser um dicionário de palavras que podem ser retiradas do texto por serem comuns, ou seja, um dicionário de *stop-words*. Essa classe é utilizada pela classe *Controller*.

3.3.1.2 Classe: Controller

A classe *Controller* é o cérebro do sistema, nela constam os algoritmos especificados na seção de modelagem. O método *processAta()* é chamado pela classe *View* para processar uma ata, nele o processo segue a ordem descrita na Figura 3.12. Ele recebe duas *Strings* com o diretório e o nome da ata, e então chama *getRawFile()* que retorna uma *String* com o texto da ata. Após segue a sequência: *findTopicList()*, *findAllContent()*, *verifySubTopics()* e *summarize-*

Content(). E então retorna uma lista de *Topics*, que é a ata hierarquizada.

O processo inicia com a chamada do método *findTopicList()* que implementa o algoritmo 2 e retorna uma lista de *Topic*, que no momento contém apenas os títulos de cada tópico. Após isso o método *findAllContent()* é chamado, ele implementa o algoritmo 3. Dentro dele 3 laços de repetição principais são efetuados.

O primeiro tem a função de encontrar as posições iniciais de cada conteúdo no texto, para isso o método *findContent()* é invocado, o qual implementa o algoritmo 3. O segundo laço define a posição final de cada conteúdo, baseado na lista de posições iniciais recebida antes. O terceiro laço extraí do texto o conteúdo de cada tópico e coloca no *content* de cada *Topic*.

Após isso tem-se uma lista de *Topic* com títulos e conteúdo, então o método *verifySubTopics()* é chamado. Nele é verificada a existência de subtópicos em cada conteúdo. Essa verificação é feita com 3 métodos, são eles: *findSubTopicsACG()*, *findSubTopicsPIEC()* e *verifySubTopicsNumberChar()*. Os quais respectivamente implementam os algoritmos 4, 5 e 6.

Por último, o método *summarizeContent()* é invocado para verificar se um conteúdo necessita ser sumarizado. E então o método *processAta()* retorna uma lista de *Topic* para a classe *View*.

3.3.1.3 Classe: View

A classe *View* é a responsável pela interface visual do sistema. Ela foi feita com a API *swing* do java e com as ferramentas da IDE *netbeans*. Nela existem 3 elementos principais: *comboBox, loadButton, treeView*. No *comboBox* são listadas todas as atas localizadas em uma pasta pré-determinada em código. O *loadButton* é usado para acionar a ação de hierarquizar uma ata. No *treeView* é apresentada a ata hierarquizada. Este componente permite expandir e retrair os nós de uma árvore, ou seja, expandir e retrair os subtópicos.

Nessa mesma classe existem dois métodos que são os principais para a apresentação da informação. O primeiro é o construtor da classe, no qual o nome das atas são carregados ao chamar o método *getFileNames* da classe *Controller*, e assim o *comboBox* é carregado com esses nomes. O segundo método é o de ação do *loadButton*. Esse método chama o método da classe Controle que processa uma ata e retornam uma lista de *Topico*. Com essa lista armazenada, o método preenche a *treeView*, que é atualizada na tela.

4 RESULTADOS

Este capítulo apresenta os resultados deste trabalho. Nele são mostrados capturas de tela do sistema *desktop* desenvolvido e também os experimentos realizados para averiguação da eficiência do algoritmo.

4.1 Interface visual

A Figura 4.1 mostra uma captura da tela do sistema. Ele é bem simples, contendo apenas uma caixa de seleção de atas, a qual lista todas as atas de uma pasta pré-especificada, um botão para carregar uma ata, e um painel que mostra o resultado da ata hierarquizada.

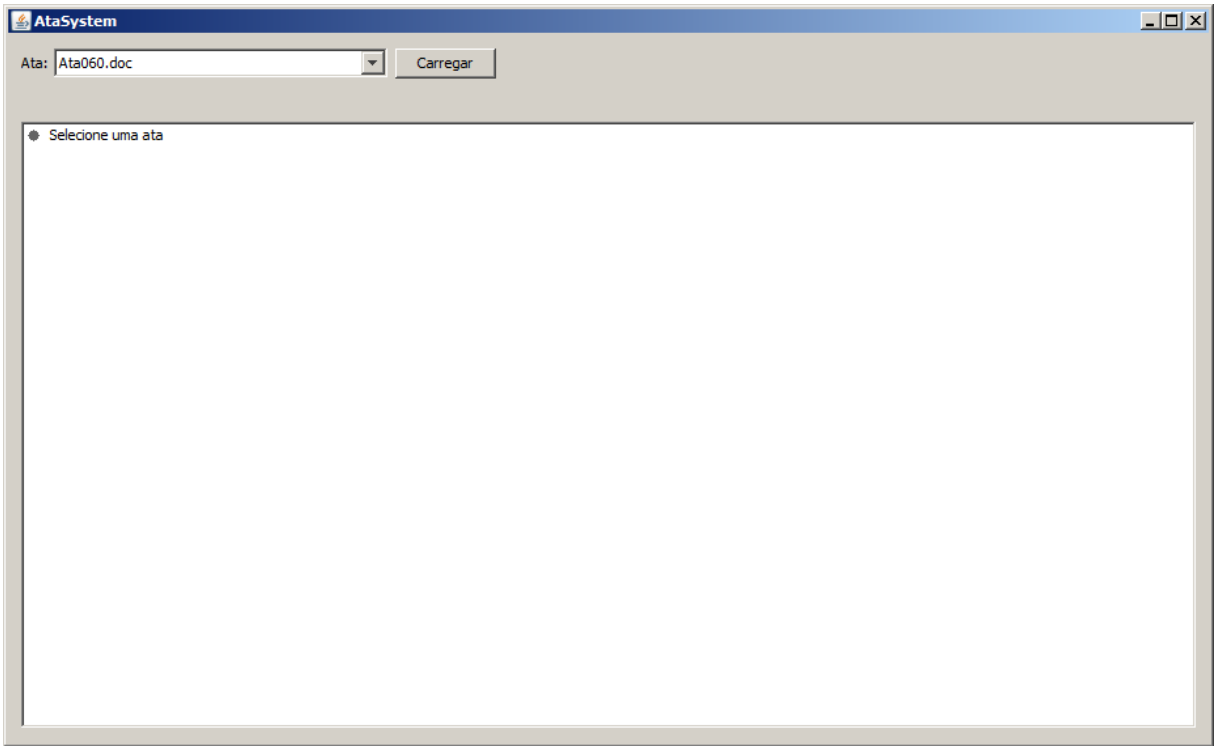

Figura 4.1 – Visão geral do Sistema Desktop

Na Figura 4.2 é possível observar uma ata já hierarquizada. Os tópicos encontrados podem ser expandidos ou retraídos, deixando assim a navegação mais limpa.

A Figura 4.3 mostra uma ata com alguns tópicos expandidos. É possível ver o tópico de Apreciação de ACGs, com os subtópicos alunos, cada aluno contendo as ACGs como subtópicos, e cada ACG tendo uma deliberação como subtópico. O outro tipo de subtópico mostrado é o de apreciação de PIECs, nele encontram-se os subtópicos alunos, cada aluno contendo o PIEC requisitado como subtópico, e o PIEC tendo a deliberação como seu subtópico.

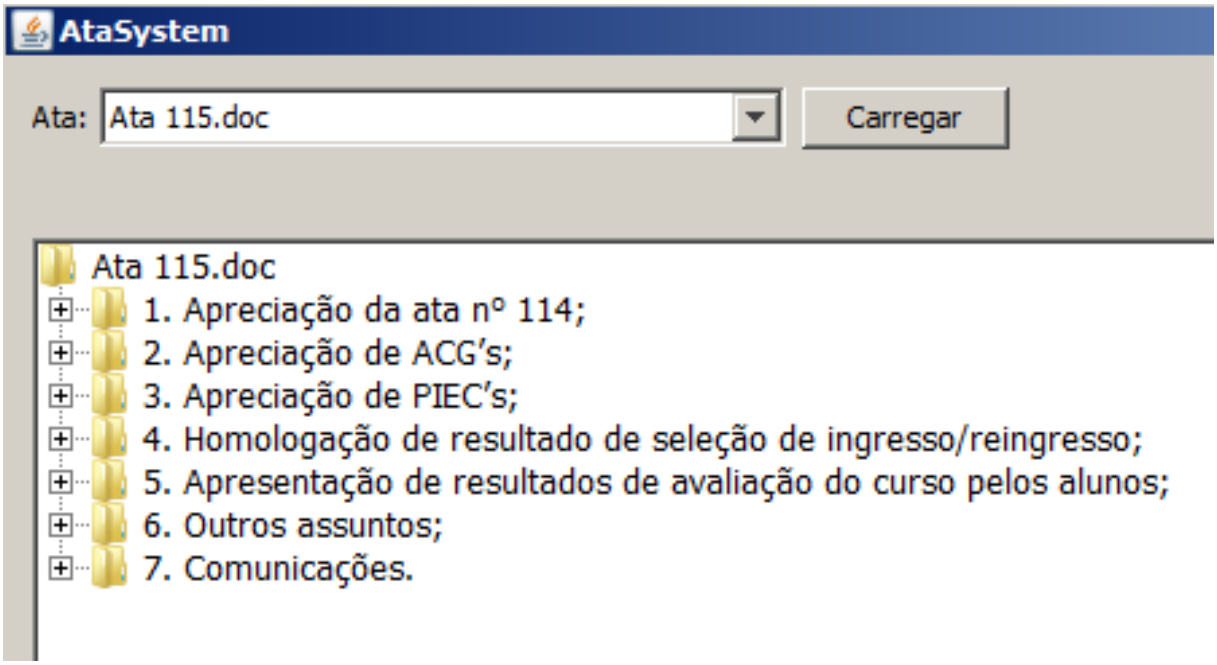

Figura 4.2 – Sistema Desktop com uma ata carregada

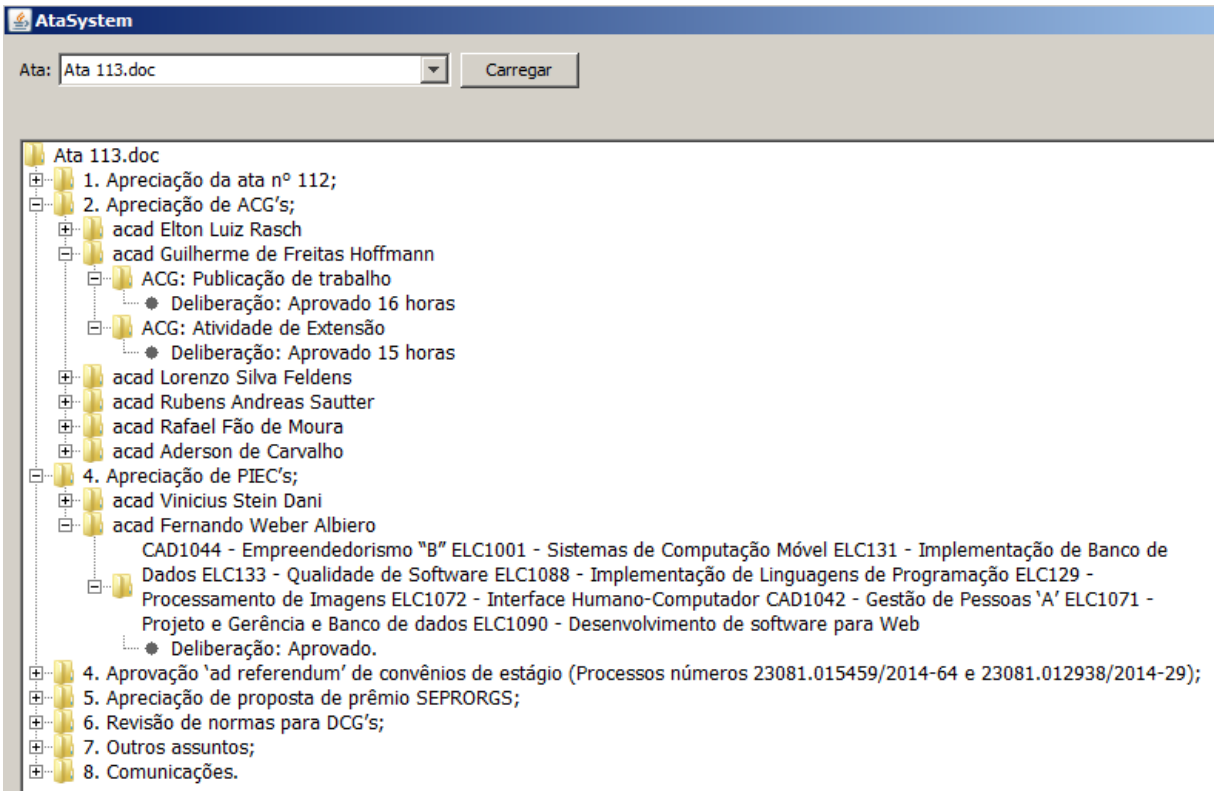

Figura 4.3 – Sistema Desktop com uma ata carregada e alguns tópicos expandidos

Nessa mesma Figura, é possível notar que há dois itens 4, isso não é um erro do sistema, mas sim um erro de digitação da ata, ele exemplifica uma das exceções que o algoritmo contorna, nesse caso o relator escreveu na ordem do dia o item 3 sendo "Apreciação

de PIEC's"e o item 4 como "Aprovação 'ad referendum' de convênios de estágio (Processos números 23081.015459/2014-64 e 23081.012938/2014-29)", porém na discussão dos itens ele referenciou ambos como itens 4.

A Figura 4.4 mostra uma ata com outro tipo de subtópico definido anteriormente, os subtópicos do tipo *número+letra*, cada item aparece em nó o qual é possível expandir ou retrair.

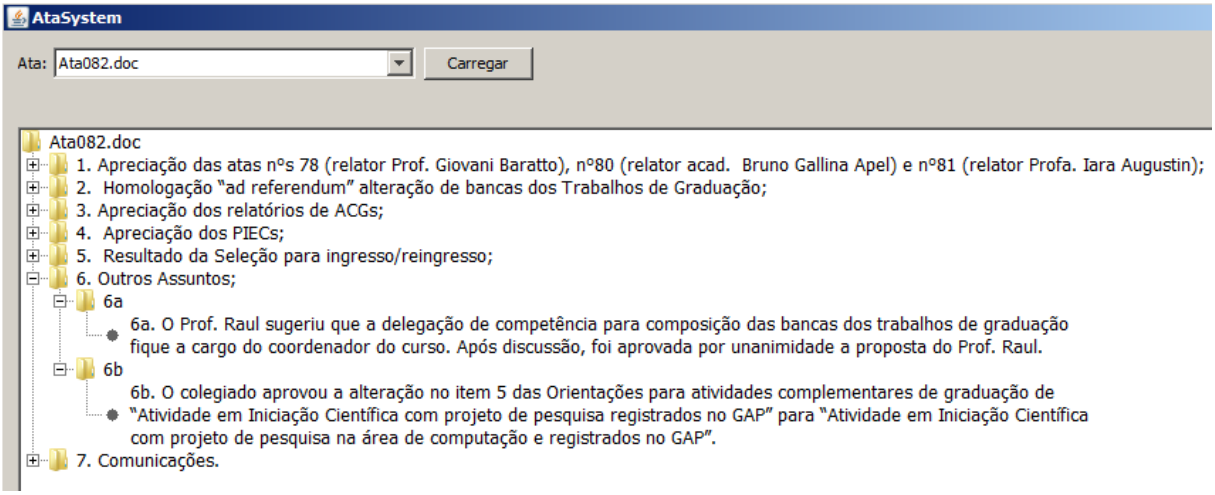

Figura 4.4 – Sistema Desktop com uma ata carregada e subtópicos do tipo número mais letra expandidos

4.2 Experimentos

Os experimentos realizados procuram demonstrar o nível de eficiência das técnicas desenvolvidas, assim como mostrar a taxa de compressão média e mostrar casos em que as técnicas não obtém sucesso. Foram testadas 76 atas, destas as técnicas funcionaram corretamente para 50, para as 26 que não obtiveram sucesso, diferentes problemas foram encontrados e são descritos abaixo.

Como mencionado, 50 atas foram corretamente manipuladas pelas técnicas, ou seja, todos os tópicos e subtópicos foram encontrados, e o conteúdo que precisou ser sumarizado, realmente o foi. Das 26 restantes, 6 possuiam erros do relator, ou seja, erros de digitação, confusão na ordem dos tópicos, etc. Com esse tipo de erro o algoritmo não encontrava os padrões definidos e assim apresentava um resultado final anormal, como o conteúdo de dois tópicos apresentados como se fossem de apenas um, ou no caso das ACGs e PIECS algumas requisições acabavam sendo ignoradas por conta dessas fugas dos padrões. A Figura 4.5 exemplifica algum desses erros, destacando-os em vermelho.

intitulado "Interface Gráfica para um Software de Processamento de Imagens Digitais, coordenado pelo prof. Marcos C. Ornellas. **Delibeação**: Aprovado 256 horas. Acad. **Luzandro Candido Tietbohl** solicitou as seguintes ACGs: **Participação em eventos**: como ouvinte: III Escola Microeletrônica da SBC Sul. **Deliberação**: Aprovado 40 horas.

reuniu-se o colegiado do Curso de Ciência da Computação, em uma sessão ordinária, com a seguinte ordem do dia: **1-Comunicações;2-Aprovação da Ata nº 040/2002;3-Apreciação "Ad referendum" bancas para as Defesas Preliminares dos Trabalhos de Graduação do 1ºsemestre/2002;4-Aprovação das bancas para as Defesas dos Trabalhos de Graduação do 1º Semestre/2002;5–Aprovação de Planos Individuais de Estudos Complementares 8-Aprovação dos Relatórios de Atividade Complementar de Graduação;6-Processo 23081.010455/2002-56–Convênio entre CACC e a UFSM;7- Outros Assuntos.**Presentes os professores Benhur de

remanejo de salas.Colocando que no cômputo geral da avaliação ficou dentro do esperado.**2.Apreciação das atas nºs 52 e 53/2004**:O Presidente colocou para apreciação e aprovação as atas nºs 52 e 53/2004.**Deliberação**:As atas foram aprovadas com as correções sugeridas pelos conselheiros.**4.Apreciação de PIECs:**Acad.**Grasiela Peccini** solicitou as seguintes disciplinas como

dia.**1.Comunicações:2.Apreciação das atas nºs 52 e 53/2004;3.Apreciação PIECs;4.Apreciação de relatórios de ACGs;5.Apreciação de alterações nas normas ACGs(relator Profa.Oni);Apreciação de alterações nas normas de DCGs(relator Prof. Benhur);7.Apreciação de alterações nas normas de TGs (relator Prof. Raul);8.Outros assuntos**.Presentes os professores André Luiz Aita, Benhur de Oliveira Stein, Marcelo Pasin, Marcos Cordeiro d'Ornellas, Oni Reasilvia de Almeida Oliveira Sichonany, Raul Ceretta Nunes e o acadêmico Rodrigo Alves Madruga representante dos alunos.O Presidente abriu a sessão e colocou para apreciação a pauta. O conselheiro Rodrigo solicitou que fosse incluído na pauta a forma de transição dos alunos para projeto político pedagógico do curso.Em seguida o presidente passou para as comunicações:**1a.**O

Figura 4.5 – Exemplo de erros em atas do colegiado do Curso de Ciência da Computação da UFSM

Outras 20 atas foram erronaemente processadas pois no meio delas foram adicionadas mais tópicos e/ou a ordem dos tópicos foi modificada. A Figura 4.6 mostra alguns casos, nela é possível observar a adição de novos tópicos e modificação de ordem sem padrão aparente, cada ata tem uma modificação diferente, dessa forma não foi possível trabalhar em uma técnica focada em um padrão de adição e modificação de ordem, porém a técnica suporta remoção de tópicos. Também é necessário dizer que em alguns casos a adição ou modificação não era feita explicitamente, apenas na parte do conteúdo o tópico aparecia sem ser previamente dito, portanto poderia se encaixar em um erro do relator.

Sichonany, Raul Ceretta Nunes e o acadêmico Rodrigo Alves Madruga representante dos alunos. O Presidente abriu a sessão e colocou para apreciação a pauta. O prof. Benhur pediu que fosse incluido na pauta o problema de gerenciamento do NCC e o Presidente sugeriu que se retirasse da pauta o item nº 9. Aprovada as alterações na pauta,

acadêmico Rodrigo Alves Madruga representante dos alunos. O Presidente abriu a sessão e solicitou a inclusão de dois itens na pauta, ficando o item 8 desdobrado em: **8.Convênio de Estágio ser firmado entre a UFSM e a Ello Assessoria em Recursos Humanos Ltda. 9.Numeração dos artigos nas normas do processo eleitoral de representante docente no colegiado do curso. 10.Outros Assuntos: 10.1.Comunicações.** Em apreciação, a

alteração de pauta. O presidente solicitou as seguintes inclusões: "**Apreciação do pedido de alteração de data de defesa preliminar do Trabalho de Graduação do acadêmico Vinicius Michel Gottin**" como item 5 e "**Inclusão da disciplina complementar de graduação ELC1070 – Gerência do Projeto de Softwar**e" como item 6. Questionou

Figura 4.6 – Exemplo de diferentes alterações na ordem dos tópicos em atas do colegiado do Curso de Ciência da Computação da UFSM

Para o teste de taxa de compressão, ou seja, o quanto as atas foram resumidas, foram usadas apenas as atas onde o algoritmo funcionou perfeitamente, totalizando 50. A Figura 4.7 apresenta um gráfico que representa o quanto essas atas foram resumidas, no qual as barras

em azul referem-se ao número de palavras antes do processo de hierarquização, e as barras vermelhas ao número de palavras depois desse processo.

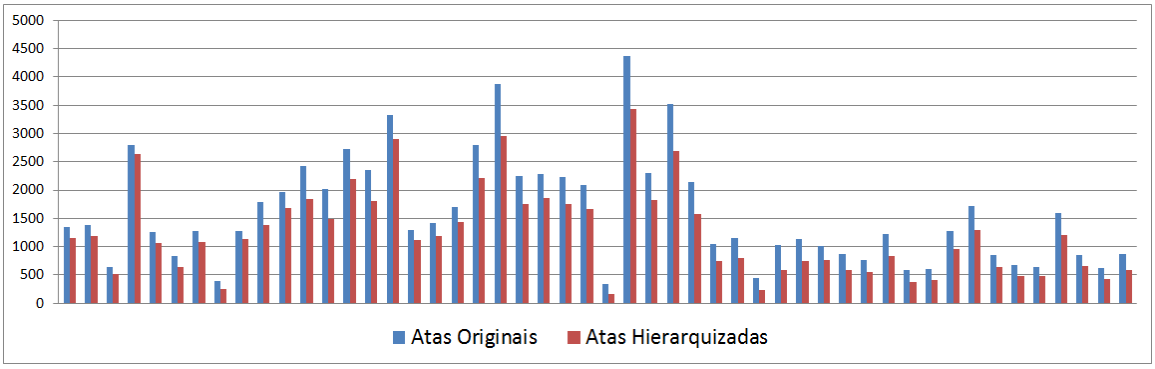

Figura 4.7 – Gráfico de diferença entre as atas originais e após a hierarquização

Em média, as atas foram resumidas em 26%. Na Figura 4.8 é possível notar que essa redução de conteúdo varia em um espectro entre 10% e 35%. Isso refere-se a dois fatores: informações não relevantes e sumarização. No primeiro caso informações como data da reunião, local, presidência, apresentação dos presentes, sempre são removidos do apresentação final, pois o foco é mostrar apenas os tópicos e seus conteúdos. O segundo fator que contribuí para a redução de informação é o processo de sumarização, pois nele os tópicos que possuem grande conteúdo tem seu volume diminuído.

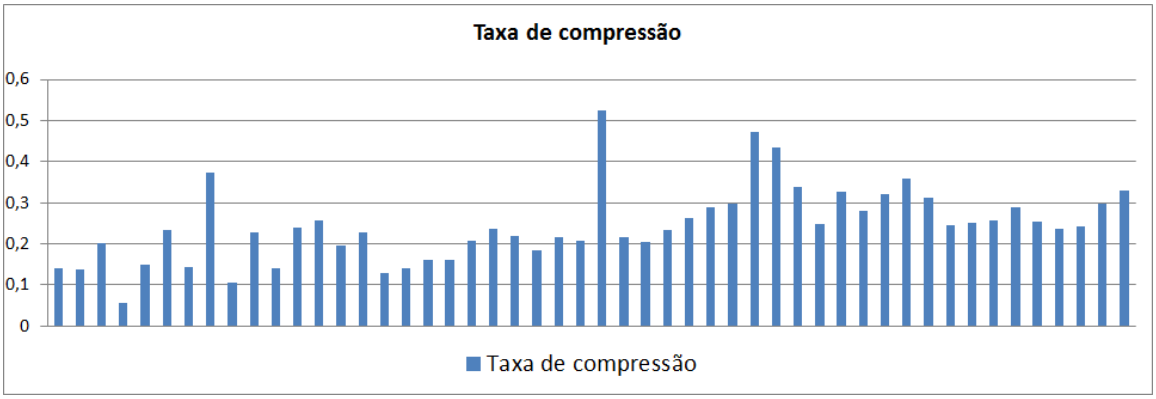

Figura 4.8 – Gráfico de taxa de compressão

Os pontos acima de 35% são atas com pouco conteúdo, a Figura 4.9 mostra um exemplo, nela apenas o trecho em azul permanece no resultado final. Nesse caso, o primeiro fator mencionado anteriormente, sobre informações como a data, local, lista de presentes e formalismos indicando a abertura da sessão, apreciação da pauta e início da ordem do dia, constituem uma grande porção do texto, dessa forma, quando são removidos, contribuem para essa alta porcentagem de compressão.

Além disso, outro fator que contribui para essa alta taxa de compressão é a própia listagem de tópicos, como seus nomes repetem no início de cada conteúdo, a lista em si é excluída na apresentação final. Dessa forma quando isso é aplicado em uma ata como a do exemplo, a qual não possuí extenso volume de palavras, também acaba sendo uma relevante porcentagem de texto comprimido.

Ata nº 81/2008

Aos vinte e sete dias do mês de novembro do ano de dois mil e oito, às dezesseis horas e trinta minutos, na sala nº 336 do Centro de Tecnologia, sob a presidência do Professor Antonio Marcos de Oliveira Candia, coordenador do Curso de Ciência da Computação, reuniu-se o colegiado do curso de Ciência da Computação, em uma sessão ordinária, com a seguinte ordem do dia: **1.Apreciação do PPC do curso de Sistemas de Informação; 2.Apreciação do PPC referente a reforma curricular do curso de Ciência da Computação.** Presentes os professores Andréa Schwertner Charão, Antonio Marcos de Oliveira Candia, Benhur de Oliveira Stein, Giovani Baratto, Iara Augustin, Oni Reasilvia de Almeida O. Sichonany, Raul Ceretta Nunes, o acadêmico Bruno Gallina Apel representante dos alunos. O presidente abriu a sessão e em seguida, colocou para apreciação a pauta da reunião. **Deliberação:** aprovada por unanimidade. Passou-se então à ordem do dia. **1.Apreciação do PPC do curso de Sistemas de Informação.** A comissão responsável pelo projeto do Curso de Sistemas de Informação apresentou o PPC do curso o qual prevê o início para o 2° semestre de 2009 com uma turma de 40 vagas. Após discussão, o presidente colocou para apreciação a aprovação do PPC do novo curso. **Deliberação**: Aprovado por unanimidade. **2.Apreciação do PPC referente a reforma curricular do curso de Ciência da Computação.** O presidente apresentou a proposta de reforma curricular no curso de Ciência da Computação a qual foi amplamente discutida com os professores do Curso. A proposta implica em aumento de 10 vagas no curso e alteração/atualização de algumas disciplinas. Em apreciação a proposta do presidente. **Deliberação**: Aprovado por unanimidade. Nada mais havendo a constar foi a presente sessão encerrada, tendo eu Janice Vendruscolo, secretária da Coordenação do Curso de Ciência da Computação, lavrado a presente Ata que vai ser assinada pelo presidente e por mim.

Prof. Antonio Marcos de Oliveira Candia,
Presidente do Colegiado.

Figura 4.9 – Exemplo de ata destacando o resultado final mostrado no sistema desktop

5 CONCLUSÃO

Este trabalho apresentou um algoritmo que, aplicado ao nicho das reuniões colegiado do curso de Ciência da Computação da UFSM, hierarquiza automaticamente atas em tópicos e subtópicos, assim como sumariza o conteúdo destes. Para apresentação e navegação da informação estruturada, foi desenvolvido um sistema *desktop*.

Durante o desenvolvimento do algoritmo, notou-se uma grande dificuldade em encontrar padrões viáveis de estruturação das atas. Algumas fugiam a qualquer padrão, e outras variavam. Portanto este foi o maior desafio deste processo.

O trabalho deixa espaço para melhorias, tanto no algoritmo, quanto no sistema desktop. No algoritmo há espaço para incluir mais padrões de estruturação, fazendo assim o algoritmo ter mais nichos de aplicação.

Quanto ao sistema desktop, é possível evoluí-los em diversos aspectos, como exemplo, adicionar um campo de busca de termos na ata, ou busca por tópicos semelhantes em diferentes atas. Também é possível planejar a evolução para um sistema web, onde possa ser acessado de qualquer navegador e atas possam ser adicionadas e excluídas da base de dados eficientemente.

Além disso, é possível adaptar os algoritmos para criar um validador de formatação de atas, dessa forma garantindo a padronização das atas desde sua escrita. Outra ideia é desenvolver um gerador de atas, também partindo da ideia de padronizar as atas desde sua criação.

Enfim, o trabalho apresentou um resultado satisfatório quanto análise de padrões, estruturação, apresentação e navegação das informações, e deixa aberto um espaço para evolução em diversos pontos.

REFERÊNCIAS

DICIO. **Ata - Dicionário Online de Português**. Acessado em: 28-03-2016, http://www.dicio.com.br/ata/.

GOULARTE, F. B. et al. Método fuzzy para a sumarização automática de texto com base em um modelo extrativo (FSumm). , [S.l.], 2015.

MERGEN, S. L. S.; HEUSER, C. A. Carla: uma técnica para comparação de cadeias de caractéres. **I Escola Regional de Banco de Dados, Porto Alegre**, [S.l.], 2005.

PARDO, T. A. S. Gistsumm: um sumarizador automático baseado na idéia principal de textos. **Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional, São Paulo**, [S.l.], 2002.

RINO, L. H. M.; PARDO, T. A. S. A Sumarização Automática de textos: principais características e metodologias. In: XXIII CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Anais...** [S.l.: s.n.], 2003. v.8, p.203–245.

SANTOS, Â. F. d. S. dos. **Sumarização automática de texto**. 2012. Tese (Doutorado em Ciência da Computação) — UNIVERSIDADE DA BEIRA INTERIOR.