UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REGRAS DE ASSOCIAÇÃO E CORRELAÇÃO TEMPORAL PARA POPULAR E DETECTAR INCONSISTÊNCIAS EM GRANDES BASES DE CONHECIMENTO

RAFAEL GARCIA LEONEL MIANI

ORIENTADOR: PROF. DR. ESTEVAM RAFAEL HRUSCHKA JUNIOR

São Carlos – SP

Dezembro/2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

REGRAS DE ASSOCIAÇÃO E CORRELAÇÃO TEMPORAL PARA POPULAR E DETECTAR INCONSISTÊNCIAS EM GRANDES BASES DE CONHECIMENTO

RAFAEL GARCIA LEONEL MIANI

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Inteligência Artificial

Orientador: Prof. Dr. Estevam Rafael Hruschka Junior

São Carlos – SP

Dezembro/2017

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a defesa de Dissertação de Mestrado do(a) candidato(a) **Rafael Garcia Leonel Miani**, realizada em **20 de Dezembro de 2017.**

****

Prof(a). Dr(a). Estevam Rafael Hruschka Junior
(UFSCar)

Prof(a). Dr(a). Heloisa de Arruda Camargo
(UFSCar)

Prof(a). Dr(a). Marilde Terezinha Prado Santos
(UFSCar)

****

Prof(a). Dr(a). Ana Carolina Lorena
(UNIFESP)

****

Prof(a). Dr(a). Alexandre Gonçalves Evzukof
(UFRG)

Certifico que a defesa realizou-se com a participação à distância dos membros **Ana Carolina Lorena, Estevam Rafael Hruschka Junior** e **Alexandre Gonçalves Evzukof** e depois das arguições e deliberações realizadas, os participantes à distância estão de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Prof(a). Dr(a). Heloisa de Arruda Camargo
(presidente

A Deus, a minha esposa Thaís, a minha filha Marina e a minha mãe Lucineide.

AGRADECIMENTOS

Agradeço a Deus pela força e persistência que me proporcionou durante o projeto. Agradeço a minha esposa Thaís e a minha filha Marina pelo apoio e suporte para o desenvolvimento do trabalho. Ao meu orientador e aos colegas de grupo pelas dicas e sugestões para o bom desenvolvimento do projeto.

RESUMO

Grandes bases de conhecimento crescente têm sido um interessante campo em muitas pesquisas nos últimos anos. A maioria das técnicas focam na construção de algoritmos para auxiliar a Base de Conhecimento (BC) a expandir automaticamente (ou semiautomaticamente). Entretanto, muitas ferramentas utilizadas para a expandir as BCs podem extrair dados incompletos ou incorretos, tornando a base inconsistente. Dessa forma, este trabalho possui o objetivo de expandir as grandes bases de conhecimento e detectar inconsistências nas mesmas. Para tal, são utilizadas a mineração de regras de associação e a correlação temporal. Ao aplicar um algoritmo de extração de regras de associação em grandes bases de conhecimento, é necessário considerar o problema de valores ausentes, uma vez que elas crescem diariamente, não possuindo todos os dados. Logo, foi criado um novo parâmetro para realizar o cálculo do suporte, denominado MSC, para trabalhar com valores ausentes. Além disso, um grande problema ao utilizar regras de associação é o esforço gasto ao avaliar cada regra extraída. Dessa forma, o presente trabalho desenvolveu o componente ER, o qual elimina regras de associação redundantes e irrelevantes. Cada regra válida é utilizada pelo componente TARE com o objetivo de detectar inconsistências. TARE introduz o conceito de STARs (regras de associação temporais específicas), as quais são utilizadas para detectar possíveis inconsistências. Cada STAR considerada relevante é utilizada como entrada para o componente TCI com o intuito de obter correlações temporais para (i) detectar possíveis inconsistências e (ii) auxiliar a popular a BC. Experimentos realizados demonstraram que as regras de associação e a correlação temporal são capazes de expandir a base de conhecimento, diminuindo a quantidade de valores ausentes. Além disso, ambos os componentes TARE e TCI foram eficientes no processo para detectar possíveis inconsistências na base de dados. Por fim, o componente ER reduziu em mais de 30% o número de regras sem perda no processo de popular a BC.

Palavras-chave: Regras de Associação, Grandes Bases de Conhecimento, Regras de Associação Temporais Específicas, Correlação Temporal, Detecção de Inconsistência, Regras Redundantes, Regras Irrelevantes

ABSTRACT

Large growing knowledge bases have been an interesting field in many researches in the past few years. Most techniques focus on constructing algorithms to help a Knowledge Base (KB) automatically (or semi automatically) expands. However, many tools used to expand the KBs can extract incomplete or incorrect data, turning the KB inconsistent. In this way, this work has the objective to expand large knowledge bases as well as detect inconsistencies on them. To accomplish that, an association rule mining algorithm and temporal correlation are used. Applying an algorithm to extract association rules in large knowledge bases, the missing value problem need to be considered, once these bases grow day to day, and do not have all of the data. Therefore, a new parameter was created to perform the support calculation, the MSC parameter, to deal with missing values. Besides, a major problem on using association rules is the effort spent to analyze each extracted rule. Thus, this work developed ER component, which eliminates redundant and irrelevant association rules. Each valid rule is used by TARE component with the purpose of detecting inconsistencies. TARE introduces the concept of STARs (specific temporal association rules), which are used to detect possible inconsistencies. Each relevant STAR is used as an input to TCI component in order to get temporal correlations to (i) detect possible inconsistencies and (ii) to help populating the KB. Experiments showed that the association rules and the temporal correlation are capable to expand the knowledge base, decreasing the amount of missing values. Moreover, both TARE and TCI components were efficient in the process of detecting possible inconsistencies in the data set. Finally, the ER component reduced the number of rules in more then 30% without any lost in the process of populating the KB.

Keywords: Association Rules, Large Knowledge Bases, Specific Temporal Association Rules, Temporal Correlation, Inconsistency Detection, Redundant Rules, Irrelevant Rules

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

Bases de conhecimento crescentes estão sendo vastamente exploradas nos últimos anos (GALÁRRAGA et al., 2013). Muitos sistemas foram construídos com o objetivo de formar essas Bases de Conhecimento (BCs) . Cyc (MATUSZEK et al., 2006), DBpedia (BIZER et al., 2009), NELL (CARLSON et al., 2010a) e YAGO (SUCHANEK; KASNECI; WEIKUM, 2007) são exemplos de tais sistemas. Existem dois modos principais para estender grandes BCs: (i) construir algoritmos que buscam por dados na Web ou (ii) desenvolver ferramentas que expandem essas bases a partir de relações descobertas em cima dos fatos e categorias já armazenados nesses sistemas.

Com o intuito de estender e popular bases de conhecimento, diferentes técnicas e ferramentas foram desenvolvidas. Algumas com o objetivo de estender suas instâncias, populando sua BC com elas, e outras para criar novas categorias e relações entre as categorias da base de conhecimento inicial.

Dentre os vários sistemas acima citados, o NELL é o de principal interesse neste projeto de pesquisa. O NELL (Never-Ending Language Learning) é um sistema computacional que trabalha 24 (vinte e quatro) horas por dia, 7 (sete) dias por semana, extraindo informações a partir de textos na internet para popular e ampliar sua própria base de conhecimento. Os principais objetivos desse sistema são aprender a ler melhor a web diariamente e armazenar o conhecimento adquirido em uma BC crescente que nunca para de evoluir. O NELL utiliza muitos componentes, como CPL (CARLSON et al., 2009), CSEAL (CARLSON et al., 2010b), Prophet (APPEL; HRUSCHKA, 2011), OntExt (MOHAMED; HRUSCHKA JUNIOR; MITCHELL, 2011), e Conversing Learning (PEDRO; HRUSCHKA JUNIOR, 2012b), que foram desenvolvidos não apenas para expandir sua BC, mas para avaliar o conhecimento obtido e evitar a propagação de erros.

Sua base de conhecimento é representada por uma ontologia, caracterizada por categorias (ou domínios), pelas relações entre elas e por suas instâncias.

Considerando que a BC do NELL cresce diariamente, ela não possui todos os valores de cada categoria, tampouco as relações entre todas as categorias descritas na ontologia, uma vez que os componentes do NELL descobrem novos fatos e relações a cada dia. No entanto, algumas relações podem nunca ocorrer para instâncias específicas. Por exemplo, imagine as categorias *cidade* e *universidade*. Logo, pode ocorrer uma relação *cidadeTemUniversidade(X, Y)*. Porém, nem toda cidade possui uma universidade. Assim, uma instância desta relação, contendo um par de argumentos (cidade e universidade) específicos pode nunca ocorrer. Na realidade, a maioria dos pares (cidade, universidade) possíveis não irão ocorrer (uma universidade não está na maioria das cidades). Desse modo, a base de conhecimento do NELL, quando analisada como uma base de dados tradicional (arquivo texto em que cada linha representa uma tupla, e cada coluna um atributo), contém muitos valores ausentes (*missing values*), ou seja valores que ainda não estão presentes para determinados atributos em uma tupla. Quando se utiliza de técnicas baseadas em itens frequentes para extrair regras de uma base consistindo de tal propriedade, tem-se duas opções: i) explorar novas técnicas utilizando algoritmos de conjunto de itens frequentes; ii) usar técnicas de extração de regras relacionais (como o trabalho de Gardner et al. (2013), por exemplo).

Outro problema a ser considerado nessas bases de conhecimento (como o NELL, por exemplo) é a identificação do período em que um determinado evento ocorreu. Por exemplo, considere a relação *pessoaPresidentePaís(Getúlio Vargas, Brasil)*, a qual representa que a pessoa *Getúlio Vargas* é presidente do *Brasil*. Para os dias atuais, a relação estaria com informação incorreta. Contudo, se for adicionado o escopo temporal para a mesma, a sentença pode se tornar válida, como *pessoaPresidentePaís(Getúlio Vargas, Brasil), 1930-1945*, a qual indica que *Getúlio Vargas* foi presidente do *Brasil* nesse período.

Grandes bases de conhecimento crescente podem ter armazenado dados que não estão totalmente corretos e que foram aprendidos erroneamente, resultando em um conjunto de dados com algumas inconsistências. Isso se deve ao aprendizado incorreto obtido pelos algoritmos empregados para expandir suas bases. O dado incorreto pode ocorrer por vários motivos, como a presença de homônimos. Por exemplo, a instância *São Paulo* pode exemplificar um estado, uma cidade, um santo, um time de futebol, entre outros. Dependendo do contexto em que foi armazenada, pode resultar em dados inconsistentes, levando à propagação de erros em futuros aprendizados.

1.2 Objetivo

Dessa forma, este trabalho de doutorado teve como principais objetivos:

- Utilizar técnicas de extração de regras de associação para auxiliar a popular e estender a base de conhecimento do NELL, explorando as características da presença de valores ausentes existentes na base e;
- Utilizar regras de associação temporais específicas e de correlação temporal para (i) popular a base de conhecimento do NELL e (ii) identificar inconsistências na mesma.

1.3 Metodologia e Contribuições

A primeira parte deste projeto consiste em popular a base de conhecimento do NELL com dados utilizando a mineração de Regras de Associação (RA) (AGRAWAL; IMIELINSKI; SWAMI, 1993). O algoritmo NARFO (MIANI et al., 2009) foi utilizado e modificado para atender aos objetivos do projeto e à característica da BC. O algoritmo NARFO foi escolhido por ser capaz de navegar em uma estrutura ontológica ou taxonômica, reconhecendo a qual categoria pertence determinado fato, e por ser capaz de trabalhar com Regras de Associação Generalizadas (RAG) (SRIKANT; AGRAWAL, 1995), que são úteis para identificar as categorias envolvidas em uma regra de associação (o que é utilizado por outro componente no processo de detecção de inconsistências).

Como citado previamente, uma base de conhecimento como a do NELL possui muitos valores ausentes. Existem dois principais motivos que explicam a presença de células vazias (valores ausentes) em uma base de conhecimento grande e crescente como o NELL. O primeiro está relacionado às instâncias que ainda não foram extraídas pelos algoritmos do NELL, isto é, elas não estavam disponíveis no momento em que o algoritmo de extração de regras de associação foi executado, mas podem estar disponíveis no futuro. O segundo motivo consiste na relação específica entre duas categorias, que pode estar definida na ontologia, mas nunca ocorrer. Isso será melhor explicado nas seções posteriores. Dessa forma, ao se aplicar um algoritmo de mineração de regras de associação em tais bases é necessário tratar desse problema. Para isso, o cálculo do suporte dos itemsets foi modificado, resultando em um novo parâmetro para o cálculo do suporte, denominado de MSC (Modified Support Calculation). Resumidamente, a medida MSC descarta um itemset durante o cálculo do suporte se todos os itens envolvendo as categorias do itemset estão ausentes. A criação desse parâmetro é uma das contribuições

deste trabalho e os resultados foram publicados em (MIANI; PEDRO; HRUSCHKA JUNIOR, 2014) e (MIANI; HRUSCHKA JUNIOR, 2015).

Além disso, devido à característica da BC, é necessário avaliar quais regras são válidas e podem ser usadas para popular a base com instâncias. Ao contrário do que ocorre com algoritmos tradicionais de extração de regras de associação, que consideram regras fortes aquelas que possuem as medidas de suporte e confiança maior ou igual às mínimas predefinidas, as regras descobertas que possuem alto valor para esses parâmetros, podem trazer falsos padrões, ou seja, que não são verdadeiros. Isso ocorre devido ao fato do NELL não possuir todas as instâncias e por crescer a cada dia. Por consequência, regras com suporte e confiança maiores que os desejados devem ser cuidadosamente analisadas antes de serem consideradas úteis.

Outro problema a ser considerado quando se trabalha com algoritmos de RAs é o esforço gasto no processo de análise das regras extraídas. Com o intuito de aprimorar o processo de avaliação dos padrões descobertos e amenizar a análise das regras, foi desenvolvido o componente ER (Eliminating Rules). O ER trabalha com a eliminação de dois tipos de regras de associação:

1. Super-regras antecedentes consideradas redundantes;
2. Super-regras consequentes consideradas irrelevantes.

O primeiro item introduz o conceito de *super-regras antecedentes* (super antecedent rules - super ARs), que são regras que possuem o mesmo consequente, mas o antecedente de uma delas é um superconjunto. Já o segundo item introduz as *super-regras consequentes* (super consequent rules - super CRs), que são regras com o mesmo antecedente, sendo o consequente de uma delas é um superconjunto. Ao aplicar esses procedimentos, o esforço para analisar as regras diminui, a cada iteração, uma vez que todas as técnicas têm o propósito de facilitar a avaliação das regras, diminuindo a quantidade de regras geradas. Deve-se ratificar que o algoritmo é executado em uma grande base de conhecimento com crescimento contínuo. Neste sentido, sem um tratamento para reduzir o número de regras, a execução do algoritmo e o processo de análise das regras obtidas se torna cada vez mais árduo.

Uma regra de associação é considerada relevante se foi descoberta e utilizada para expandir a BC, populando-a com instâncias ou a incrementando com novas relações entre as categorias do NELL. Caso contrário, a regra é irrelevante e não pode ser utilizada no processo com o intuito de evitar a propagação de erros. Os experimentos realizados mostraram que a aplicação do componente ER diminui o número de regras geradas em mais de 30%, mostrando sua necessidade e eficiência. O ER é outra contribuição deste trabalho de doutorado.

Tabela 1.1: Exemplo de regras de associação e regras de associação temporais específicas.

Número	Regra	Data inicial	Data final
1	*atletaGanhaTroféu(X, super_bowl) → atletaJogaEsporte(X, Futebol_Americano)*		
2	*atletaGanhaTroféu(ben_roethlisberger, super_bowl) → atletaJogaEsporte(ben_roethlisberger, Futebol_Americano)*	2004	2014
3	*atletaGanhaTroféu(eli_manning, super_bowl) → atletaJogaEsporte(eli_manning, Futebol_Americano)*	2004	2014
4	*atletaGanhaTroféu(favre, super_bowl) → atletaJogaEsporte(favre, Futebol_Americano)*	2008	2008
5	*atletaGanhaTroféu(kurt_warner, super_bowl) → atletaJogaEsporte(kurt_warner, Futebol_Americano)*	2000	2000
6	*atletaGanhaTroféu(peyton_manning, super_bowl) → atletaJogaEsporte(peyton_manning, Futebol_Americano)*	2007	2007

Duas técnicas foram desenvolvidas com a finalidade de detectar possíveis inconsistências. A primeira técnica utiliza o componente TARE (Temporal Association Rules Extraction), o qual utiliza as regras de associação válidas extraídas. O TARE possui duas principais contribuições no processo de detecção de inconsistências:

1. Introdução das regras de associação temporais específicas (Specific Temporal Association Rules - STARs), que possui resultados em (MIANI; HRUSCHKA JUNIOR, 2017b);

2. Criação de duas métricas probabilísticas para detecção de inconsistências (MIANI; HRUSCHKA JUNIOR, 2017a).

Cada regra de associação gerada é formada por um conjunto de STARs. Uma STAR possui uma data inicial e final. Neste trabalho, as datas iniciais e finais são representadas pelos anos iniciais e finais. Na tabela 1.1, a regra número 1 é uma regra de associação gerada pelo algoritmo e é utilizada para popular a BC do NELL. A regra diz que se um atleta ganha o troféu *super bowl*, ele joga o esporte *futebol americano*. As regras de 2 a 6 são consideradas STARs. Cada STAR foi utilizada para gerar a regra da linha 1. As STARs aparecem com uma data inicial e final. Note que, para as linhas 2 e 3, o ano inicial é diferente do final, e, para as linhas de 4 a 6, o ano inicial é o mesmo do final. Uma STAR que possui a data inicial diferente da final é considerada *não pontual*. Caso contrário, a STAR é denominada *pontual*.

Além da introdução das STARs, o componente TARE possui outra contribuição: o uso de duas métricas probabilísticas para a verificação de possíveis inconsistências na base de conhecimento:

- Métrica Probabilística 1 (MP1), que leva em consideração a probabilidade de uma STAR ser *pontual* ou não;

- Métrica Probabilística 2 (MP1), que considera a probabilidade de uma STAR ser *pontual* ou não com base em todas as regras de associação com as mesmas categorias.

A métrica MP1 considera a probabilidade de uma STAR ser pontual ou não para uma determinada regra de associação, considerando todas as STARs utilizadas para gerar a regra. Por exemplo, nas STARs da tabela 1.1, as STARs 2 e 3 são *não pontuais*, e as STARs de 4 a 6 são *pontuais*. Logo, pela MP1, o algoritmo indica que as STARs para a regra da linha 1 possui característica pontual, e as que não forem pontuais devem ser investigadas.

Para entender como a MP2 funciona, considere a regra de associação *RA1* e as *STAR1* e *STAR2* a seguir:

RA1: *atletaGanhaTroféu(X, campeonato nba)* $\rightarrow$ *atletaJogaEsporte(X, Basquete)*

STAR1: *atletaGanhaTroféu(Lebron_James, campeonato nba)* $\rightarrow$ *atletaJogaEsporte(Lebron_James, Basquete) (2013, 2013).*

STAR2: *atletaGanhaTroféu(Kobe_Bryant, campeonato nba)* $\rightarrow$ *atletaJogaEsporte(Kobe_Bryant, Basquete) (1996, 2013).*

Pela MP1, tem-se que 50% das regras podem ser pontuais ou não pontuais. Logo, a MP1 pede auxílio à MP2 que, por sua vez, considera todas as regras de associação que envolvem essas categorias. Caso as regras sejam *pontuais*, o TARE indicará para investigar as que não forem pontuais. No exemplo, a regra 1 da tabela 1.1 possui característica pontual. Logo, o algoritmo indicará que a *STAR2* deve ser investigada.

Contudo, se a *RA1* fosse descoberta antes das regras e STARs da tabela 1.1, como seria identificada a pontualidade de uma regra? Com o intuito de evitar propagação de erros na base de dados, no primeiro ciclo de iteração é realizado um estudo com as categorias da base de dados utilizada para identificar quais domínios possuem a característica pontual e quais não possuem. Para o caso em questão, identifica-se, por exemplo, que a categoria *GanhaTroféu* possui característica *pontual*, o que sugere uma investigação para verificar as que não o são. A partir do segundo ciclo de iterações, as métricas funcionam corretamente sem a necessidade de um estudo. A definição e exemplos de ciclos de iteração serão descritas no capítulo de Arquitetura do Sistema.

Cada STAR considerada relevante é utilizada como entrada pelo componente TCI. Este

possui dois principais objetivos:

- Obter correlações temporais entre a STAR e os dados do subconjunto utilizado para atualizar a base de conhecimento do NELL;
- Obter correlações temporais entre a STAR e os dados do subconjunto utilizado para detectar inconsistências presentes na base de conhecimento do NELL.

Neste trabalho, o temo correlação temporal se refere a comparação entre uma STAR e as demais instâncias da BC que possuem uma intersecção em relação ao período temporal.

O componente TCI utiliza o conjunto de STARs e um subconjunto da BC do NELL como entrada. O TCI compara cada STAR com o subconjunto de dados correspondente para descobrir correlações temporais para atualizar a BC do NELL ou para detectar inconsistências (MIANI; HRUSCHKA JUNIOR, 2017b). Considerando a *STAR2* anterior, a qual relata que o atleta Kobe Bryant ganhou o *torneio* da *nba* e que joga o esporte *basquete*, tem-se que esse mesmo atleta possui outros fatos como *atletaJogaTime(Kobe Bryant, lakers)*. O algoritmo procura por instâncias que possuem algum valor em comum entre a instância atual (Kobe Bryant) e as instâncias do subconjunto, que tenham alguma intersecção temporal para o período do *Kobe Bryant*. Caso seja encontrado algum valor em comum, como o time que o atleta joga, o TCI atualiza a outra instância com os valores do atleta *Kobe Bryant* sempre que encontrar um valor ausente.

No entanto, caso o algoritmo encontre alguma diferença entre as instâncias da base, mas que possuam algum valor em comum durante um período específico de tempo, uma possível inconsistência é detectada. Considerando as *STAR1* e *STAR2* do exemplo anterior e imaginando agora uma situação em que a *STAR2* tenta relacionar o atleta *Kobe Bryant* com a instância correspondente à do atleta *Lebron James*, tem-se que ambos os atletas possuem um valor em comum para a categoria *troféu* e uma intersecção com relação as suas datas iniciais e finais. Suponha agora que o atleta *Lebron James* jogue no time *Miami Heat*. Neste caso, existe uma diferença entre os fatos para uma mesma categoria, mesmo ambas as instâncias possuindo algum valor em comum, o que sugere uma possível inconsistência em uma ou nas duas instâncias.

Antes de executar o componente TCI é necessário realizar um estudo para identificar quais instâncias são relevantes para serem utilizadas na consulta para a correlação temporal com o intuito de atualizar a BC. Imagine que o único valor entre duas instâncias seja o esporte praticado, por exemplo. Isso não implica que joguem no mesmo time ou tenham o mesmo técnico. Portanto, esse estudo também é necessário para evitar a propagação de erros e atualizar a BC corretamente.

Para realizar a correlação temporal, o TCI adapta a ideia de intervalo algébrico temporal de Allen (TIA - Temporal Interval Algebra) (ALLEN, 1983), o qual possui 13 relações contando com as reversas. Neste trabalho, foram utilizadas 11 relações, adaptando algumas de Allen TIA para os métodos aqui utilizados (MIANI; HRUSCHKA JUNIOR, 2017b). O TIA de Allen será melhor descrito no capítulo de Temporalidade, e a adaptação realizada neste trabalho no capítulo de Arquitetura do Sistema.

Dessa forma, com o propósito de popular a base de conhecimento do NELL e detectar inconsistência na mesma, o presente trabalho possui as seguintes contribuições:

1. O uso de regras de associação para auxiliar a popular uma grande base de conhecimento;
2. A criação do parâmetro MSC para algoritmos de regras de associação em uma grande base de conhecimento crescente para tratar dos valores ausentes;
3. A necessidade de analisar cada regra de associação descoberta;
4. O desenvolvimento de um novo método para eliminação de regras de associação redundantes;
5. O desenvolvimento de um novo método para eliminação de regras de associação irrelevantes;
6. A introdução do conceito de regras de associação temporais específicas (STARs);
7. A avaliação das regras de associação temporais específicas extraídas por meio de métricas probabilísticas para detectar inconsistências na base de dados;
8. A adaptação do intervalo algébrico temporal de Allen para realizar a atualização e a detecção de inconsistências na base de dados;
9. Um novo método para detectar inconsistências com base na correlação temporal;
10. A atualização da BC baseada na correlação temporal.

As contribuições acima tiveram alguns resultados aceitos e publicados em conferências e periódicos, listados a seguir:

- **Association Rules to Help Populating a Never-Ending Growing Knowledge Base**. Lecture Notes in Computer Science. 1ed.: Springer International Publishing, 2014, v. 8864, p. 169-181 - IBERAMIA - 2014;

- **Analyzing the use of obvious and generalized association rules in a large knowledge base**. In: 2014 14th International Conference on Hybrid Intelligent Systems (HIS), 2014, Kuwait. 2014 14th International Conference on Hybrid Intelligent Systems, 2014. p. 1-6;

- **Exploring Association Rules in a Large Growing Knowledge Base**. International Journal of Computer Information Systems and Industrial Management Applications, v. 7, p. 106-114, 2015;

- **Specific temporal association rules and temporal correlations to enlarge and detect inconsistencies in a large growing knowledge base**. In: 2017 13th ICNC-FSKD, 2017, China. IEEE, 2017, p. 1537-1546;

- **Correcting inconsistencies trough association rules in temporal large knowledge bases**. In: 2017 6th BRACIS, 2017, Brasil, p. 1-6;

- **Eliminating Redundant and Irrelevant Association Rules in Large Knowledge Bases**. In: 2018 20th ICEIS, 2018, Portugal, a ser publicado em março de 2018.

1.4 Estrutura do trabalho

O capítulo 2 descreve Regras de Associação e algumas de suas variantes usadas neste trabalho. Grandes Bases de Conhecimento, bem como alguns sistemas e algumas técnicas utilizados para expandir-las são apresentados no capítulo 3. Além disso, ainda no capítulo 3, o NELL e BCs inconsistentes serão melhores detalhados. O capítulo 4 traz a questão dos valores ausentes: (i) como ocorrem em grandes bases de conhecimento e (ii) como algoritmos de regras de associação trabalham com tal problema. Em seguida, no capítulo 5, são descritos dados temporais e como estão sendo utilizados em diversos problemas. A arquitetura do sistema com todas as contribuições deste trabalho são descritas no capítulo 6. Os experimentos, os resultados obtidos e os trabalhos comparativos são mostrados no capítulo 7.

Capítulo 2

REGRAS DE ASSOCIAÇÃO

2.1 Definição

Considere-se I = $(i_1, i_2, ..., i_n)$ o conjunto de itens de um banco de dados D.

Definição 1 *Uma Regra de Associação (RA) consiste em uma implicação $X \rightarrow Y$, em que $X \subseteq I$, $Y \subseteq I$, e $X \cap Y = \theta$ (AGRAWAL; IMIELINSKI; SWAMI, 1993) .*

O antecedente da regra é representado pelo *X*, e o consequente pelo *Y*. Cada regra de associação possui duas medidas: suporte e confiança. O suporte de uma regra se refere à porcentagem de transações em D que contêm *X e Y*, o que é calculado através da equação 2.1 (na equação, ocorrências (X ∪ Y) refletem o número de transações em que X e Y ocorrem juntas na base de dados D, e T é o número total de transações). O parâmetro de confiança é a porcentagem de transações contendo X que também contém Y, sendo calculada pela equação 2.2. Tradicionalmente, uma regra de associação é considerada forte se seus respectivos graus de suporte e confiança forem maiores ou iguais às medidas chamadas de *minsup* (suporte mínimo desejado) e *minconf* (confiança mínima desejada).

$$suporte(X \rightarrow Y) = \frac{ocorrências(X \cup Y)}{T} \tag{2.1}$$

$$confiança(X \rightarrow Y) = \frac{suporte(X \cup Y)}{suporte(X)} \tag{2.2}$$

Um exemplo de tal regra seria a relação *pão* $\Longrightarrow$ *presunto, queijo*, contendo grau de 3% (0.03) de suporte e 80% (0.8) de confiança. Isso significa que em 3% de todas as transações da base de dados, os itens *pão, presunto e queijo* aparecem juntos. Por outro lado, uma regra

com 80% de confiança permite dizer que em 80% das vezes em que aparece o item *pão* em uma transação, também ocorreu a presença dos itens *presunto e queijo*. Uma regra é considerada forte (ou relevante) se o seu grau de suporte e confiança são iguais ou maiores às medidas de mínimo suporte (*minsup*) e confiança (*minsup*) desejados, respectivamente. Consequentemente, uma regra de associação caracteriza o quanto a presença de um item na base de dados pode implicar na presença de um outro item ou conjunto de itens na mesma.

Definição 2 *Um itemset representa um conjunto de itens em um banco de dados D.*

O suporte de um *itemset* é a porcentagem de transações que possui todos os itens do *itemset*. Um itemset é considerado frequente se o valor de seu suporte é maior ou igual ao *minsup*.

Resumidamente, a tarefa de regras de associação consiste em duas etapas principais:

1. Encontrar todos os *itemsets* frequentes;
2. Gerar regras de associação fortes a partir dos *itemsets* frequentes.

Diversos são os algoritmos existentes na literatura para trabalhar com extração de regras de associação. O algoritmo *Apriori* (AGRAWAL; SRIKANT et al., 1994) é, provavelmente, o mais amplamente conhecido e utilizado algoritmo em mineração de regras de associação. A partir dele, vários outros foram desenvolvidos, com diferentes propósitos, como o FP-Growth (HAN; PEI; YIN, 2000), o DHP (PARK; CHEN; YU, 1995), o DLG (YEN; CHEN, 1996), e o Max-Miner (JR, 1998), por exemplo.

O objetivo do algoritmo *Apriori* é identificar um conjunto de *itemsets* frequentes e construir regras de associação a partir desse conjunto. As RAs que possuírem suporte e confiança maior ou igual que *minsup* e *minconf*, respectivamente, são consideradas fortes e relevantes. Esse algoritmo trabalha com a ideia de que se um conjunto não é frequente, ou seja, se não possui suporte maior ou igual ao mínimo desejado, todos os seus super conjuntos são descartados, o que resulta em melhor performance.

O algoritmo *Apriori* trabalha em um processo iterativo, em que *k-itemsets* (itemsets de tamanho k) são utilizados para encontrar *(k+1) itemsets*. Primeiramente, o conjunto de *1-itemset* é encontrado. Todos os que forem frequentes (tiverem o grau de suporte maior ou igual ao *minsup*) são utilizados para encontrar os itemsets de tamanho 2. O processo continua até não existirem mais *k-itemsets* frequentes. Considere-se L_1, L_2, ..., L_k, conjuntos de itemsets denominados *1-itemset*, *2-itemset*, ..., e *k-itemset*, respectivamente. Ao procurar por cada L_k, uma

Algoritmo 1: Algoritmo Apriori

```
1  L_1 = Conjuntos de itemsets frequentes de tamanho 1;
2  para (k = 2; L_{k-1} ≠ ∅; k++) faça
3  |  C_k = apriori-gen(L_{k-1}); – geração de candidatos
4  |  para todas as transações t faça
5  |  |  C_t = subset(C_k, t); – candidatos contidos na transação t
6  |  |  para todos os candidatos c em C_t faça
7  |  |  |  c.contagem++;
8  |  |  fim
9  |  fim
10 |  L_k = {c em C_k | c.contagem ≥ minsup}
11 fim
12 Resultado = reunião de todos L_k
```

varredura completa é realizada na base de dados. No final do processo, tem-se o conjunto de itemsets frequentes que será utilizado para formar as regras de associação. As regras que possuírem os graus de suporte e confiança maiores ou iguais ao suporte e à confiança mínimos desejados, serão consideradas válidas.

O *Apriori* é apresentado no algoritmo 1. L_k representa o conjunto de itens frequentes, enquanto C_k corresponde aos *itemsets* candidatos. Cada *itemset* de C_k que for frequente é adicionado a L_k.

O algoritmo *Apriori* pode ser dividido em duas etapas:

1. Geração de *itemsets* candidatos: busca e identificação de cada *itemset* frequente;
2. Geração de regras de associação: geração de regras a partir de cada *itemset* frequente.

Na etapa de geração de candidatos, a função *apriori-gen* (linha 3 do Algoritmo 1) recebe como parâmetro L_{k-1} (conjunto de todos os *itemsets* frequentes de tamanho k-1) e retorna todos os *itemsets* frequentes de tamanho k. Essa função, normalmente, pode ser dividida em dois passos: *Join* (Junção) e *Prune*(Poda). Os Algoritmos 2 e 3 descrevem esses passos.

No passo *Join* são realizadas combinações através da junção de L_{k-1} com L_{k-1}. Ou seja, são realizadas junções dos *itemsets* frequentes de tamanho *k-1* para obter os candidatos de tamanho k (C_k). No passo da poda (*Prune*), são eliminados os itemsets candidatos que possuem subconjuntos não frequentes. Isto é, um ajuste é realizado em que todos os itens $c \in C_k$ são removidos, de tal forma que qualquer subitem c de tamanho *k-1* não esteja em L_{k-1}.

Após a geração de todos os *itemsets* L_k, o algoritmo está apto para a etapa de geração das regras de associação. Considerando I como um *itemset* frequente do conjunto L_k, uma regra de

Algoritmo 2: Passo *Join* - adaptado de (AGRAWAL; SRIKANT et al., 1994)

insere em C
selecione p.item$_1$, p.item$_2$, ..., p.item$_{k-1}$, q.item$_{k-1}$
de L$_{k-1}$ p, L$_{k-1}$ q
onde p.item$_1$ = q.item$_1$, ..., p.item$_{k-2}$ = q.item$_{k-2}$, p.item$_{k-1}$ < q.item$_{k-1}$;

Algoritmo 3: Passo *Prune* - adaptado de (AGRAWAL; SRIKANT et al., 1994)

```
para Para todos itemsets frequentes c ∈ C_k faça
    para todos (k-1) subconjuntos s de c faça
        se (s ∉ L_{k-1}) então
            apague c de C_k;
        fim
    fim
fim
```

Tabela 2.1: Exemplo de uma base de dados amostral D

TID	Lista de itens
T100	I1, I2, I5
T200	I2, I3, I4
T300	I3, I4
T400	I1, I2, I3, I4

associação pode ser apresentada como (I - a) $\rightarrow a$, em que $a \subseteq I$. Para a regra ser considerada válida, como já mencionado, o valor de confiança da regra tem que ser maior ou igual ao mínimo desejado. A confiança é calculada conforme a equação 2.2. No exemplo dado, X = (I - a) e Y = a.

Para melhor exemplificar esse procedimento, tem-se a base de dados amostral representada pela tabela 2.1. A figura 2.1 ilustra a geração dos itemsets frequentes da tabela 2.1. Em cada iteração k (em que k é o número da iteração em questão), o algoritmo verifica quais k-itemsets possuem suporte maior ou igual ao mínimo definido. O processo começa com a procura por itemsets candidatos de tamanho 1 (C$_1$). Considere-se que o suporte mínimo seja igual a 0.5, ou seja, a quantidade de vezes que um itemset aparece na amostra, dividido pelo total de transações tem que ser maior ou igual a 0.5. Dessa forma, o conjunto L$_1$ pode ser obtido. Para obter o conjunto C$_2$, o algoritmo faz a junção dos 1-itemsets frequentes. Os 2-itemsets candidatos que tiverem suporte maior ou igual a 0.5 são considerados frequentes. A partir deles, é obtido o conjunto (C$_3$). O processo continua até que não seja obtido nenhum k-itemset frequente. No exemplo em questão, C$_4$ não possui nenhum *itemset* candidato.

Com todos os itemsets frequentes encontrados, entra a fase de geração das regras. Aqui, todos os *k-itemsets* frequentes já têm seu valor de suporte calculado. As regras que tiverem grau de confiança maior ou igual ao mínimo predefinido são consideradas fortes e serão exibidas

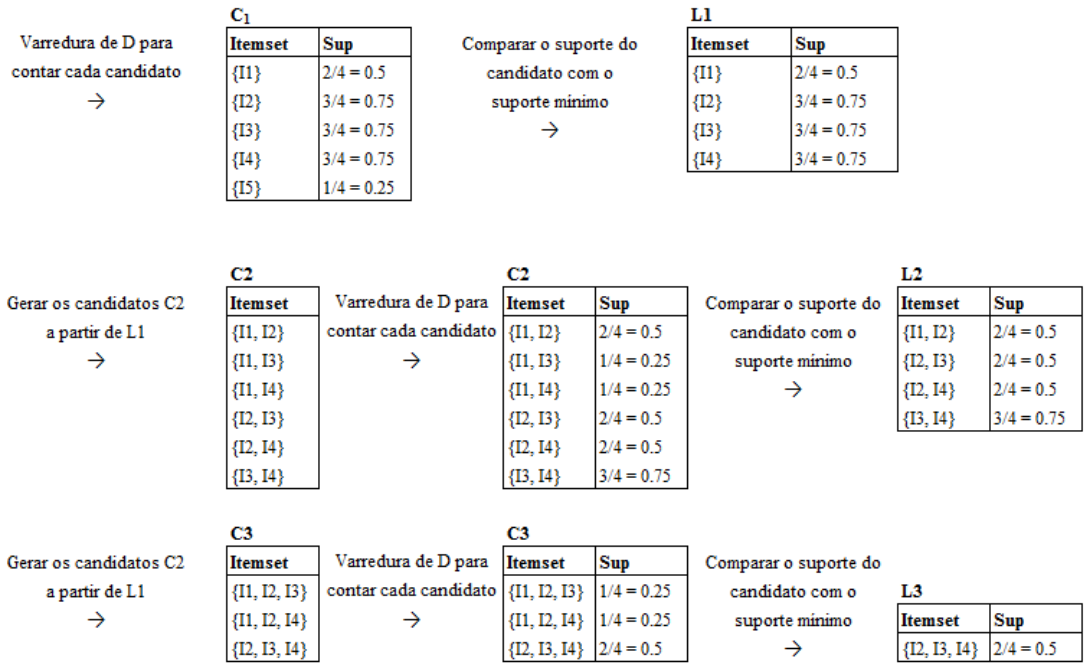

Figura 2.1: Exemplo de geração de itemsets frequentes

Tabela 2.2: Exemplo de regras geradas

Regra	Suporte	Confiança
I2 → I3, I4	0.5	2/3 ≈ 0.67
I3 → I2, I4	0.5	2/3 ≈ 0.67
I4 → I2, I3	0.5	2/3 ≈ 0.67
I2, I3 → I4	0.5	2/2 ≈ 1
I2, I4 → I3	0.5	2/2 ≈ 1
I3, I4 → I2	0.5	2/3 ≈ 0.67

no final do algoritmo. Para cada *itemset* frequente, o algoritmo gera todas as possibilidades de regras da forma *antecedente* → *consequente*. Aquelas que possuírem o grau de confiança maior ou igual ao *minconf* são consideradas válidas.

Considerando o *itemset* frequente I2, I3, I4, tem-se que a tabela 2.2 mostra todas as possibilidades de regras para ele, com seus respectivos graus de suporte e confiança. Com o valor do *minconf* fixado em 0.7, apenas duas regras seriam mostradas ao final da iteração do algoritmo.

De acordo com Han, Kamber e Pei (2006), o algoritmo *Apriori* é o mais aplicado na mineração de conjuntos de itens frequentes para Regras de Associação. Esse algoritmo representou um grande diferencial em relação aos algoritmos anteriores a ele, principalmente no que se refere ao desempenho e à estratégia (WOJCIECHOWSKI; ZAKRZEWICZ, 2002). Desse modo, o *Apriori* é considerado um algoritmo clássico, sendo base para o desenvolvimento de outros algoritmos (PEREGO; ORLANDO; PALMERINI, 2001).

Com o surgimento das taxonomias (SRIKANT; AGRAWAL, 1995), das ontologias (GRUBER, 1993), do aumento das bases de conhecimento e, por consequência, da quantidade de regras

geradas, muitas sendo redundantes e/ou irrelevantes, novas técnicas foram desenvolvidas. Taxonomias e ontologias proporcionam o uso de regras de associação generalizadas (SRIKANT; AGRAWAL, 1995), uma vez que trabalham com categorias ou domínios, não somente com os fatos das instâncias específicas das categorias.

Um grande problema envolvendo regras de associação é a quantidade de regras geradas, que dificulta a escolha sobre como analisá-las. Muitas regras extraídas são consideradas redundantes. Assim, vários algoritmos foram criados com o foco de eliminar regras redundantes e diminuir a quantidade de regras geradas. Os algoritmos de extração de regras de associação podem gerar muitas regras irrelevantes. Logo, criar mecanismos para diminuir a quantidade de regras irrelevantes se torna um processo interessante.

Por fim, verificar o período em que uma determinada regra ocorreu pode trazer maior confiabilidade para ela. Por exemplo, considere-se a regra *atletaGanhaTroféu(Tim_Duncan, campeonato nba)* $\rightarrow$ *atletaJogaLiga(Tim_Duncan, nba)*, a qual indica que se o atleta em questão ganhou a liga nba, então ele participou dessa liga. No entanto, se o ano em que o mesmo ganhou for conhecido, a mesma terá maior acurácia e veracidade. Portanto, além da descrição de regras de associação, este trabalho também descreve regras de associação generalizadas, redundantes, irrelevantes e temporais, que são utilizadas no desenvolvimento deste trabalho de doutorado. A mineração de regras de associação temporais será melhor abordada no capítulo sobre temporalidade.

2.2 Regras de Associação Generalizadas

Em muitas ocasiões, os fatos das instâncias pertencem a determinadas categorias ou domínios. Isto é, o valor *nba* pertence a uma liga esportiva, *fanta* é um tipo de refrigerante, e *UFSCar* e *USP* são exemplos de universidades, por exemplo. Dessa forma, taxonomias (*hierarquia "é um"(is-a hierarchy)*) podem ser muito úteis.

Considere-se I = $i_1, i_2, ..., i_n$ o conjunto de itens de um banco de dados D e T uma taxonomia (ou hierarquia "é um"). T também é conhecido como grafo acíclico direcionado (directed acyclic graph). Uma aresta em T representa um relacionamento "é-um". Se existe uma aresta em T de p para c, diz-se que p é pai de c e c é filho de p (p é a generalização de c). Um item x' é um *ancestral* de x (ou x é *descendente de x'*) se existe uma aresta de x' para x em T (SRIKANT; AGRAWAL, 1995). Um exemplo de taxonomia pode ser visto na figura 2.2. Nela, tem-se uma aresta de agasalhos para jaquetas, por exemplo. Assim, *agasalhos* é pai (ou ancestral) do filho (descendente) *jaquetas*.

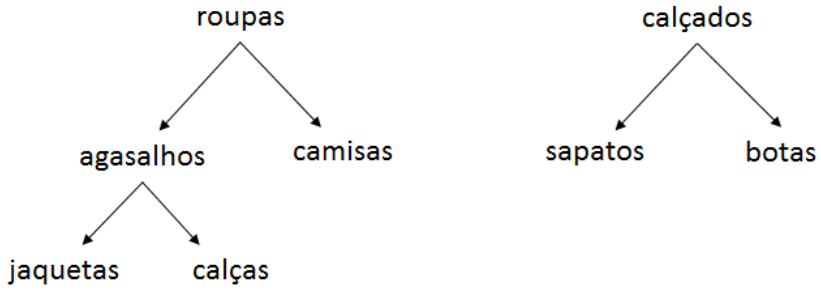

Figura 2.2: Exemplo de taxonomia - adaptado de (SRIKANT; AGRAWAL, 1995)

Definição 3 *Uma regra de associação generalizada é uma implicação da forma $X \rightarrow Y$, em que $X \subseteq I$, $Y \subseteq I$, e $X \cap Y = \emptyset$, e nenhum item em Y é ancestral de nenhum item em X (SRIKANT; AGRAWAL, 1995).*

Uma regra "$x \rightarrow$ *ancestral (x)*"é trivialmente verdadeira com 100% de confiança, o que justifica a afirmação "nenhum item em *Y* é *ancestral* de nenhum item em *X*". O cálculo do suporte e confiança é o mesmo das equações 2.1 e 2.2. Considerando a figura 2.2, regras como "*agasalhos* $\rightarrow$ *sapatos*"e "*calçados* $\rightarrow$ *camisas*"são exemplos de regras de associação generalizadas, uma vez que, pelo menos no antecedente ou no descendente, um dos itens foi generalizado para um dos seus respectivos ancestrais.

RAGs também podem ser extraídas por meio da estrutura de ontologias. Do ponto de vista psicológico, Guarino (1998) define ontologia como sendo um sistema particular que descreve uma determinada visão de mundo. Em âmbitos computacionais, uma das definições mais utilizadas é

> "Uma ontologia é uma especificação formal e explícita de uma conceitualização compartilhada."(GRUBER, 1993, p. 2)

De acordo com Nardi, Brachman et al. (2003), a família das *Lógicas de Descrição (Description Logics - DL)* é considerada uma das mais importantes representação formal de conhecimento, formando a base para linguagens de representação de ontologias como a *Web Ontology Language* (OWL) (MCGUINNESS; HARMELEN et al., 2004). Esta é uma das linguagens mais difundidas por ser uma recomendação oficial do consórcio *World Wide Web Consortium* (W3C) para a criação de ontologias na *Web Semântica* (BERNERS-LEE et al., 2001), sendo uma família de três linguagens (GRAU et al., 2008):

- *OWL Lite*: provê os elementos básicos para a representação de conceitos, relacionamentos e restrição simples de propriedades, sendo o dialeto OWL menos expressivo e complexo.

Em função de suas construções mais simples, facilita o desenvolvimento de ferramentas e máquinas de inferência que manipulam as ontologias. Não permite os conceitos de operações como união e complemento;

- *OWL DL*: provê expressividade aliada à garantia de que todas as inferências sejam computáveis e processadas em um tempo finito. Corresponde à Lógica de Descrição *SHOIN* (FIKES; HAYES; HORROCKS, 2004), sendo uma das linguagens mais utilizadas para a criação de ontologias na *Web* Semântica;

- *OWL Full*: é a variante da OWL mais expressiva, sem garantias computacionais.

Alguns trabalhos que utilizam ontologias como apoio no processo de mineração de RAGs podem ser encontrados em Brisson, Collard e Pasquier (2005), Miani et al. (2009), Benites e Sapozhnikova (2013) e GalÁrraga et al. (2013). No primeiro, desenvolveu-se o algoritmo ExCIS, que usa o conhecimento de domínio tanto no pré-processamento quanto no pós-processamento. A primeira tarefa utiliza a ontologia para guiar a construção de dados específicos para tarefas de mineração de dados. Na etapa de pós-processamento as regras geradas podem ser interpretadas ou filtradas, uma vez que seus termos são generalizados de acordo com a ontologia. Já em Miani et al. (2009), um algoritmo de extração de regras de associação generalizadas (denominado NARFO) é desenvolvido, e a ontologia é utilizada como apoio no processo de descoberta de padrões. NARFO possui um novo parâmetro para generalização de seus itens, chamado *minGen* (minimal generalization), em que um item de uma regra é generalizada se $x\%$ dos itens do domínio (porcentagem definida pelo *minGen*) aparece em outras regras com os mesmos antecedentes e consequentes. Em Benites e Sapozhnikova (2013), regras de associação generalizadas são utilizadas para tentar descobrir novos padrões entre categorias de duas ontologias biológicas distintas. Por fim, GalÁrraga et al. (2013) desenvolveram AMIE, um algoritmo para mineração de regras de associação generalizadas que trabalha com evidências incompletas em bases de conhecimento ontológicas.

2.3 Regras de Associação Redundantes

Algoritmos de regras de associação normalmente geram uma grande quantidade de regras para serem analisadas, dependendo da base de dados e dos parâmetros definidos. De acordo com Zaki (2000), quanto maior for o conjunto de itemsets frequentes, mais regras são mostradas para os usuários, muitas delas redundantes.

Várias pesquisas foram realizadas com o intuito de minimizar o impacto ao avaliar a quantidade de regras geradas no final do processo. Regras de associação redundantes são regras geradas que possuem significado igual ou similar a outra regra produzida pelo algoritmo. Considere-se a taxonomia da figura 2.2. Caso as regras *jaquetas* $\rightarrow$ *botas*, *camisas* $\rightarrow$ *sapatos*, *roupas* $\rightarrow$ *calçados* fossem extraídas, as duas primeiras seriam redundantes, uma vez que, neste caso, as regras específicas estariam inclusas na regra generalizada.

Com o intuito de eliminar itemsets redundantes, muitas técnicas foram desenvolvidas, como *Closed Itemsets*, e *Maximal itemsets*, por exemplo. Alguns algoritmos que produzem *closed itemsets* também realizam a geração de regras de associação, a partir dos *closed itemsets* frequentes. No entanto, não é conhecido nenhum algoritmo de mineração de *maximal itemsets* que faça a geração de regras devido ao custo computacional de verificar todo o conjunto de dados novamente para obter os suportes dos itemsets não maximais da regra.

Um *closed itemset* frequente (*Frequent Closed Itemset* - FCI) é um itemset frequente (*Frequent Itemset* - FI) X tal que não exista nenhum superconjunto de X com o mesmo suporte (PASQUIER et al., 1999). Zaki (2000) desenvolveu uma técnica para tratar a redundância produzida por algoritmos tradicionais de extração de regras baseadas em itemsets frequentes, utilizando o conceito de FCIs. Resultados demonstraram que as regras diminuíram exponencialmente em comparação com algoritmos sem nenhum tratamento de redundância. CHARM é um algoritmo eficiente para mineração de itemsets frequentes. Ele enumera conjuntos fechados usando uma árvore de busca dupla *itemset-tidset*. Ele também utiliza uma abordagem rápida baseada em hash para remover quaisquer conjuntos não fechados durante o processamento (ZAKI; HSIAO, 2002). Um outro algoritmo que faz uso de FCIs é o *A-Close* (PASQUIER et al., 1999). O número de regras diminui sem perda de informação, reduzindo o custo de computação do algoritmo. *A-Close* é um dos algoritmos mais conhecidos em se tratando de mineração de FCIs.

Um *Itemset Frequente Maximal* (Maximal Frequent Itemset - MFI) é um itemset frequente X com nenhum superconjunto frequente (BURDICK; CALIMLIM; GEHRKE, 2001). De acordo com os autores, o seguinte relacionamento é mantido: MFI $\subseteq$ FCI $\subseteq$ FI. Eles também desenvolveram um algoritmo para minerar MFIs, denominado *MAFIA*, o qual integra uma variedade de ideias de algoritmos em um algoritmo prático. FPMax (GRAHNE; ZHU, 2003) utiliza uma estrutura de árvore FP (FP-tree) para armazenar a informação de frequência de toda a base de dados. Para verificar se um itemset frequente é maximal, uma outra estrutura de árvore, denominada de Árvore de Itemsets Frequentes Maximais (Maximal Frequent Itemset tree - MFI-tree), é utilizada para manter o controle de todos os MFIs.

GenMax (GOUDA; ZAKI, 2005) e *MaxMiner* (JR, 1998) são outros algoritmos bem conhe-

Tabela 2.3: Base de dados amostral para comparação entre FIs x FCIs X MFIs

TID	Lista de itens
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE

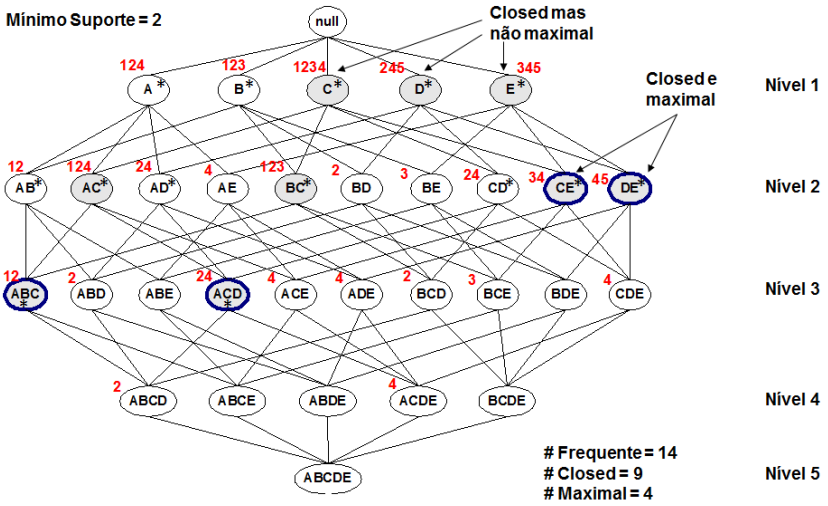

Figura 2.3: Reticulado para comparação entre FIs x FCIs x MFIs

cidos para a mineração de MFIs. *GenMax* faz uma pesquisa de retrocesso na base de dados para enumerar todos os padrões maximais com eficiência. Além disso, utiliza um número de otimizações para rapidamente eliminar uma grande porção do subconjunto amostral. *MaxMiner* foi um dos primeiros algoritmos para minerar itemsets frequentes utilizando a técnica MFI. Segundo os autores, este algoritmo tem eficácia uma vez que abandona a abordagem *bottom-up* (baixo para cima) para buscar por itemsets frequentes. Ele sempre procura mais à frente com o intuito de identificar itemsets frequentes longos. Ao identificar antecipadamente um itemset frequente longo, ele elimina todos os seus subconjuntos, não os levando em consideração.

A tabela 2.3 traz itens de uma banco de dados amostral para revelar a diferença entre a quantidade de itemsets frequentes envolvendo FIs, FCIs e MFIs. A figura 2.3 ilustra uma estrutura de um reticulado em que, a cada nível, são mostrados os possíveis itemsets de tamanho i, em que i representa o tamanho do itemset, isto é, nível 1 para itemsets de tamanho 1; nível 2, de tamanho 2, e assim por diante. Em cima de cada itemset estão os números das transações em que o itemset aparece. Por exemplo, no nível 2, o itemset BC aparece nas transações 1, 2 e 3. Para o exemplo em questão, o mínimo suporte desejado é igual a 2, ou seja, o itemset tem que aparecer em, pelo menos, duas transações.

Cada possível itemset está incluso dentro de uma elipse. FIs são identificados por um asterisco * dentro das elipses. FCIs são representados por elipses com o fundo preenchido,

enquanto os MFIs, além do fundo preenchido, também possuem a borda das elipses destacadas. Para a base amostral da tabela 2.3, tem-se um total de 14 FIs, 9 FCIs e 4 MFIs, o que mostra que a relação MFI $\subseteq$ CFI $\subseteq$ FI é verdadeira.

As técnicas de FCI e MFI são utilizadas durante a fase de geração de candidatos. No entanto, outras técnicas estão sendo utilizadas para eliminar redundância nas regras de associação extraídas na etapa de pós processamento. O algoritmo TNR (*T*op-*k* *N*on-redundant *R*ules) foi desenvolvido por Fournier-Viger e Tseng (2012). Os autores usam uma estratégia de mineração de regras de associação denominada *top-k*, em que k é o número de regras de associação a serem encontradas. No entanto, eles adicionaram um tratamento de regras redundantes ao procedimento. O algoritmo consiste em encontrar um conjunto L contendo k regras de associação, de tal modo que para cada regra $r_a \in L \mid \text{conf}(r_a) > minconf$, não existe uma regra $r_b \notin L \mid \text{conf}(r_b) > minconf \wedge sup(r_b) \geq sup(r_a)$. Caso contrário, r_b é considerada redundante em relação a r_a.

Em Marinica e Guillet (2010), uma tarefa de pós processamento foi desenvolvida para diminuir a quantidade de regras de associação. Eles propuseram um método interativo, segundo o qual especialistas do domínio da base de dados filtram as regras extraídas. CoGAR (BARALIS et al., 2012) é um algoritmo de regras de associação generalizadas que introduz duas novas medidas: (i) um esquema de restrição é criado por um analista que direciona a fase de mineração de itemsets e (ii) uma restrição de confiança oportunista que identifica regras significantes e redundantes na etapa de pós processamento.

Vo, Hong e Le (2013) criaram um novo algoritmo para tratar de regras redundantes, a partir do qual foi desenvolvido um método com base em redes (lattice-based approach) para a rápida mineração das denominadas *most generalization association rules* (MGARs) (VO; LE, 2010). Embora o método de MGARs trouxesse regras contendo menos redundância, o tempo de execução aumentava conforme o número de *itemsets frequentes fechados* (closed frequent itemsets) crescia. Assim, eles construíram um novo algoritmo para rede de itemsets frequentes fechados, assim como um teorema para eliminar nós da rede para a geração das regras.

Em Miani et al. (2009), o algoritmo NARFO foi desenvolvido excluindo algumas regras redundantes. NARFO trabalha com a generalização das regras de associação. Dessa forma, caso fossem geradas uma regra generalizada e suas específicas, apenas a regra de associação generalizada apareceria no final da iteração, uma vez que a generalização engloba os itens específicos do domínio generalizado. Logo, as regras específicas eram consideradas redundantes e eliminadas.

2.4 Regras de Associação Irrelevantes

Semelhante ao problema descrito na seção anterior, em que foram apresentados alguns trabalhos que tinham o objetivo de eliminar regras redundantes para diminuir a quantidade de regras resultantes e, consequentemente, o esforço na avaliação das mesmas, muitos algoritmos foram construídos com o objetivo de tratar Regras de Associação Óbvias (ou Triviais) e Regras de Associação Irrelevantes.

TOPSIL-Miner (YANG; HUANG, 2010) é um algoritmo que armazena itemsets potenciais em uma estrutura chamada de TOPSIL-Tree. A estratégia para redução de regras consiste em três passos: (i) a poda dos nós triviais ou óbvios do fluxo corrente de dados, (ii) proporcionar um limiar de suporte à mineração durante o processo de mineração adaptativa e heuristicamente e (iii) promover um limiar de poda dinamicamente.

O OPUS_AR (WEBB; ZHANG, 2002) reduz o espaço de busca adicionando algumas restrições nos relacionamentos entre as regras de associação com o intuito de diminuir o número de regras. O algoritmo trabalha detectando e descartando as regras triviais durante a busca, reduzindo o tempo para extrair as regras, assim como eliminando as regras óbvias das retornadas para o usuário.

Um novo parâmetro foi desenvolvido por Hashikami e Koda (2014) para amenizar a geração de grande quantidade de regras óbvias trazidas pelos algoritmos de extração de regras de associação: *Unexpectedness*. Embora os autores tenham desenvolvidos duas novas medidas em seu trabalho, apenas *Unexpectdness* possui tratamento para verificar regras triviais ou irrelevantes que seriam geradas pelo algoritmo. Esse parâmetro tem o intuito de descobrir regras inesperadas e é definido por *probabilidade condicional* e pelo coeficiente de *Sorensen-Dice*.

Segundo Burton et al. (2014), os algoritmos tradicionais de mineração de regras de associação produzem regras com muita acurácia, mas que são não apenas óbvias, mas também sem utilidade para os pesquisadores. Para resolver tal problema, eles aprimoraram os algoritmos padrões usando a técnica de *agrupamento* para identificar questões relacionadas e eliminar com antecedência regras envolvendo questões similares, uma vez que são menos propensas de serem interessantes.

Djenouri, Drias e Bendjoudi (2014) exploraram a extração de meta-regras com o intuito de eliminar regras irrelevantes. Primeiro, eles agrupam as regras de associação para grandes conjunto de dados. Depois, diferentes dependências entre as regras de um mesmo agrupamento são extraídas usando algoritmos de meta-regras, e o algoritmo de eliminação utiliza essas dependências para apagar as regras dedutivas e manter somente as regras representativas para cada

grupo. O algoritmo PVARM foi proposto por (RAMESHKUMAR; SAMBATH; RAVI, 2013). Ele usa a técnica n-cross de validação para reduzir a quantidade de regras de associação irrelevantes.

Kim (2017) afirma que a geração de muitas regras de associação (muitas irrelevantes) pode ser utilizada se o usuário puder selecionar itens relevantes para serem utilizados na mineração. Dessa forma, eles desenvolveram uma ferramenta de exploração visual, denominada SAM (structured association map), que permite aos usuários encontrar grupos de itens relevantes de uma maneira visual. Resultados mostraram que o método em questão diminuiu significantemente a complexidade na análise das regras, evitando regras de associação irrelevantes.

2.5 Considerações Finais

O presente trabalho faz uso de Regras de Associação com o intuito de popular a base de conhecimento do NELL com instâncias. Para isso, algumas das técnicas descritas neste capítulo foram utilizadas. O uso de Regras de Associação Generalizadas facilita a verificação do domínio a que um determinado item pertence para uma determinada taxonomia. Por exemplo, o item *nba* pertence ao domínio *LigaEsportiva*. Assim, sabe-se quais são todos os domínios envolvidos em uma determinada regra. A generalização também será importante para os componentes de detecção de inconsistências. Como relatado previamente, algoritmos de extração de regras com base em um conjunto de itens frequentes, geram uma grande quantidade de regras, o que aumenta consideravelmente conforme a expansão da base de dados. Por consequência, é desenvolvido o tratamento de eliminação de regras redundantes e irrelevantes no pós-processamento, com o intuito de diminuir o número de regras a serem analisadas. Este trabalho faz uso de todos os conjuntos de itemsets frequentes (FIs) uma vez que as regras de associação minimais são mais úteis para popular a base de conhecimento do NELL em comparação com os métodos de FCIs e MFIs. No entanto, o presente trabalho introduziu o parâmetro MSC, que descarta itemsets com valores ausentes. Esse parâmetro faz com que mais itemsets sejam gerados. Dessa forma, o conjunto de itemsets frequentes com valores ausentes (FIMVs - Frequent Itemsets with Missing Values) é maior que o de FIs.

Capítulo 3

GRANDES BASES DE CONHECIMENTO

3.1 Definição

O campo de *Sistemas Especialistas*, que teve seu início por volta de 1970, incorporou o paradigma de que *Conhecimento* é poder: até mesmo computadores muito rápidos requerem grandes quantidades de cada conhecimento específico para resolver problemas não triviais (GUARINO; GIARETTA; MARS, 1995). Tem-se, então, o surgimento das Grandes Bases de Conhecimento (GBC).

Desde a primeira conferência internacional em construção e compartilhamento de grandes bases de conhecimento (*First International Conference on Bulding and Sharing Very Large-Scale Knowledge Bases*) em 1993, vários sistemas foram criados, tendo grande avanço com o crescimento da Internet.

Atualmente, muitos sistemas como Cyc (MATUSZEK et al., 2006), DBpedia (BIZER et al., 2009), NELL (CARLSON et al., 2010a), YAGO (SUCHANEK; KASNECI; WEIKUM, 2007), YAGO2 (HOFFART et al., 2013) (o qual estende o YAGO para tratar da dimensão envolvendo espaço e tempo), Freebase (BOLLACKER et al., 2008a) e Knowledge Vault (DONG et al., 2014a) têm o objetivo de criar grandes bases de conhecimento procurando por fatos e relações existentes na Web, ou aplicando algoritmos para descobrir relações com base no que já foi armazenado.

3.2 Sistemas com Grandes Bases de Conhecimento

3.2.1 Cyc

Desde seu início, o objetivo principal do projeto Cyc[1] têm sido construir uma GBC contendo um armazém de conhecimento de apoio formalizado adequado para uma variedade de tarefas de raciocínio e para a solução de problemas em uma variedade de domínios.

Atualmente, Cyc possui cerca de 630.000 conceitos, aproximadamente 7 milhões de asserções (composta por fatos e regras), utilizando mais de 38.000 relações. Um subconjunto do Cyc, denominado plataforma OpenCyc, está disponível gratuitamente para ser utilizado.

O sistema Cyc possui os seguintes componentes e mecanismos que, em conjunto, auxiliam a construir sua base de conhecimento (SIEGEL et al., 2004):

- Base de Conhecimento (Knowledge Base);
- Mundos (Worlds);
- Mecanismo de Inferência (Inference Engine);
- Interfaces do Usuário (User Interfaces);
- Transcrições e o Servidor de Transcrição (Transcripts and the Transcript Server);
- Partições (Partitions);
- Componente de Integração de Fonte de Conhecimento Semântico (Semantic Knowledge Source Integration (SKSI) Facility);
- APIs.

A maioria das asserções na BC tem a finalidade de obter conhecimentos do senso comum pertencentes aos objetos e eventos da vida humana, como vendas, alimentos, construções, veículos, espaço e tempo.

3.2.2 DBpedia

O projeto DBpedia[2] (BIZER et al., 2009) é um trabalho colaborativo com o propósito de extrair conhecimento estruturado do Wikipedia e tornar essa informação acessível na Web.

[1] http://www.opencyc.org/
[2] http://wiki.dbpedia.org/

A versão em inglês da base de conhecimento do DBpedia descreve 4.58 milhões de fatos, dos quais 4.22 milhões são classificados em uma ontologia consistente, incluindo 1.445.000 pessoas, 735.000 lugares (478 mil desses estão povoados), 411.000 trabalhos criativos (123.000 discos musicais, 87.000 filmes e 19.000 vídeo games), 241.000 organizações (incluindo 58.000 companhias e 49.000 instituições educacionais), 251.000 espécies e 6.000 doenças. Além disso, eles possuem versões do DBpedia em 125 linguagens, o que totaliza um total de 38 milhões de fatos rotulados.

É interessante notar que, assim como Cyc, a base de conhecimento do DBpedia cresce continuamente, conforme adquire novos fatos e conhecimentos da Web. Também é capaz de ler definições em mais de 120 línguas.

Para cada página no Wikipedia, o DBpedia cria um código identificador, o Uniform Resource Identifier (URI) , para identificar uma entidade ou um conceito que é descrito na página correspondente no Wikipedia (MENDES; JAKOB; BIZER, 2012). Durante o processo de extração, as informações estruturadas são extraídas como triplas RDF e são adicionadas na base de conhecimento como propriedades da URI correspondente.

Para realizar a extração, existem vários componentes. Os principais são:

- *PageCollections*: são abstrações locais ou remotas de fontes de artigos do Wikipedia;
- *Destinations*: armazenam as triplas RDF extraídas;
- *Extractors*: transformam tipos específicos de marcações da wiki em triplas;
- *Parsers*: dão suporte aos extratores ao determinar tipos de dados, converter valores entre unidades diferentes e dividir as marcações de listas.

3.2.3 YAGO

YAGO[3] (SUCHANEK; KASNECI; WEIKUM, 2007) é uma grande base de conhecimento semântico. Originalmente, os fatos eram automaticamente extraídos do Wikepedia e unificados com o WordNet, usando uma cuidadosa combinação baseada em métodos heurísticos e em outras técnicas que extraem regras. Atualmente, a versão atual do YAGO utiliza um conhecimento semântico derivado do Wikipedia, WordNet e GeoNames.

Hoffart et al. (2013) apresentaram YAGO2, que é uma extensão do YAGO, em que entidades, fatos e eventos são relacionados ao tempo e ao espaço. Essa nova versão da base de

[3] www.yago-knowledge.org

conhecimento foi criada não somente a partir da Wikipedia e do WordNet, mas também do GeoNames.

YAGO trabalha com triplas RDF, denominadas pelos autores de triplas SPO (subject-property-object). Para criar o YAGO2, um novo modelo foi desenvolvido: tuplas SPOTL (SPO + Time (tempo) + Location(localização)), que pode coexistir com as triplas SPO. YAGO2 traz as seguintes contribuições:

- Um *framework extensível para a extração de fatos*, que pode procurar em infoboxes, listas, tabelas e categorias em textos, e permite uma especificação rápida e fácil de novas extrações de regras;
- Uma extensão do modelo de representação de conhecimento voltado para capturar espaço e tempo, assim como regras para a propagação das informações de espaço e tempo para todos os fatos relevantes;
- Métodos para obter fatos temporais do Wikipedia e para integrar tipos e fatos espaciais a partir do GeoNames;
- Uma nova representação *SPOTL(X)* de fatos aprimorados de espaço e tempo, com consultas expressivas e de fácil uso.

YAGO3 (MAHDISOLTANI; BIEGA; SUCHANEK, 2014) é uma extensão da base de conhecimento do YAGO que combina as informações de Wikipedias em diferentes linguagens. A técnica foi utilizada em 10 (dez) linguagens diferentes com precisão variando entre 95% e 100% no mapeamento de atributos.

YAGO possui cerca de 10 milhões de entidades (como pessoas, organizações, cidades, entre outros), com mais de 120 milhões de fatos sobre essas entidades.

3.2.4 Freebase

Freebase (BOLLACKER et al., 2008a) é um sistema de banco de dados desenvolvido para ser um repositório público de conhecimento do mundo. O Freebase tenta reunir a escalabilidade de um banco de dados estruturados com a diversidades de wikis colaborativas em um banco de dados prático e escalável de conhecimento humano comum.

De acordo com Tanon et al. (2016), o Freebase foi adquirido pela Google em 2010. No entanto, devido ao sucesso do Wikidata (VRANDEČIĆ; KRÖTZSCH, 2014), a Google anunciou

sua intenção de desligar o Freebase e ajudar a comunidade com a transferência do Freebase para o Wikidata[4]. O desligamento oficial aconteceu em agosto de 2016. Freebase continha, aproximadamente, 2.4 bilhões de fatos e 44 milhões de tópicos.

3.2.5 Knowledge Vault

Knowledge Vault (DONG et al., 2014a) é uma base de conhecimento probabilística na Web (os dados são obtidos através da análise de textos, tabelas de dados, páginas estruturadas, e anotações humanas) que combina extrações de conteúdo da Web com conhecimento prévio adquirido de repositórios de conhecimento existentes.

Assim como a maioria das BCs, o Knowledge Vault também armazena as informações na forma de triplas RDF. Por exemplo, na tripla </m/02mjmr, /people/person/place_of_birth, /m/02hrh0_>, o sujeito */m/02mjmr* é o identificador para Barack Obama, e */m/02hrh0* o identificador para Honolulu. Associado com cada tripla está um valor de confiança, que indica a probabilidade que o Knowledge Vault acredita que a tripla esteja correta, isto é, sem erros.

Knowledge Vault possui três componentes principais:

- *Extractors*: extraem triplas de uma grande variedade de fontes da Web. Cada extrator anexa um valor de confiança para a tripla extraída, mostrando a incerteza sobre a identificação de uma relação e seus argumentos correspondentes;
- *Graph-based priors*: aprendem a probabilidade apriori de cada tripla, baseado nas triplas armazenadas em BCs existentes;
- *Knowledge fusion*: computa a probabilidade de uma tripla ser verdadeira, baseado na concordância entre os extratores e os graph-based priors.

Sua base de conhecimento é aproximadamente de 1.6 bilhões de fatos, englobando mais de 4 mil diferentes tipos de relações e de 1100 diferentes tipos de entidades. Cerca de 271 milhões desses fatos têm 90% de probabilidade de serem verdadeiros.

3.2.6 NELL

NELL[5] (Never-Ending Language Learning) é um sistema computacional que opera 24 horas por dia, 7 dias na semana, extraindo informação de textos na web para popular e expandir

[4]https://www.wikidata.org
[5]http://rtw.ml.cmu.edu/rtw/

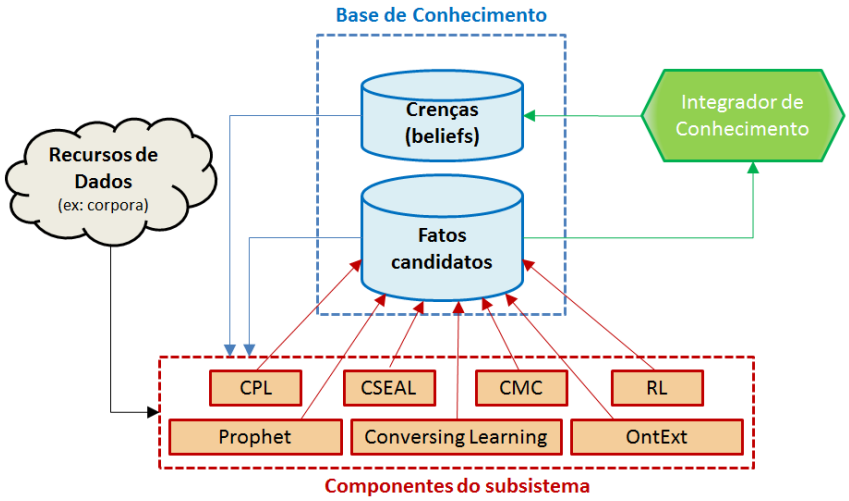

Figura 3.1: Arquitetura do NELL - adaptada de (CARLSON et al., 2010a)

sua própria base de conhecimento. O principal objetivo do sistema é aprender a ler a web melhor a cada dia e armazenar o conhecimento adquirido em uma base de conhecimento que nunca para de crescer. Para isso, o sistema faz uso de métodos de aprendizado semissupervisionado, de um conjunto variado de métodos de extração de conhecimento, e de uma representação de base de conhecimento flexível que permite a integração das saídas obtidas por esses métodos. CPL (CARLSON et al., 2009), CSEAL (CARLSON et al., 2010b), Prophet (APPEL; HRUSCHKA, 2011), OntExt (MOHAMED; HRUSCHKA JUNIOR; MITCHELL, 2011), e Conversing Learning (PEDRO; HRUSCHKA JUNIOR, 2012b) são exemplos de alguns métodos do NELL.

A base de conhecimento do NELL é representada por uma estrutura ontológica caracterizada por categorias, relacionamentos e suas instâncias. Em sua origem, algumas categorias e fatos iniciais foram definidos para a ontologia do NELL. Com base na leitura diária da Web pelos seus componentes, a base de conhecimento foi expandindo cada vez mais. Os componentes do NELL também buscam avaliar se um determinado fato aprendido é verídico ou não, afim de evitar a propagação de erros. NELL acumulou um total de mais de 50 milhões de fatos candidatos lendo a web, com diferentes níveis de confiança. Em mais de 2.407.506 dos fatos, NELL possui alta confiança.

Neste trabalho de doutorado, foi utilizada a base de conhecimento do NELL. Dessa forma, as subseções a seguir descrevem um pouco mais de sua arquitetura e seus componentes.

3.2.6.1 Arquitetura do NELL

A arquitetura do NELL é ilustrada na figura 3.1. Nela são mostrados alguns dos seus componentes utilizados para obter fatos candidatos para expandir a base de conhecimento.

O sistema é organizado em torno de uma base de conhecimento compartilhada, que cresce continuamente, e é utilizado por um conjunto de componentes de leitura/aprendizado do subsistema que implementam métodos complementares de extração de conhecimento. A base de conhecimento inicial define uma ontologia (coleção de predicados definindo categorias e relações), e alguns exemplos base para cada predicado nessa ontologia (por exemplo, uma dúzia de cidades). Os objetivos principais são aumentar de forma contínua a BC, e ler melhor a cada dia.

As instâncias de categoria e relacionamento adicionadas à BC são divididas em fatos candidatos e crenças. Os componentes do subsistema podem ler a partir da BC e consultar fontes externas (por exemplo, o corpo de textos e a Internet) e, então, sugerir novos fatos candidatos. Os componentes fornecem uma probabilidade para cada candidato e um resumo das evidências que dão suporte a ele. O *Integrador de Conhecimento* (Knowledge Integrator) examina esses novos fatos e promove os que possuem suporte forte ao estado de crença.

3.2.6.2 CPL - *Coupled Pattern Learner*

CPL é um extrator de texto livre que aprende padrões contextuais como ”prefeito de X”e ”X joga para Y”para instâncias de categorias e relações. O componente CPL utiliza estatísticas de co-ocorrências entre frases nominais e padrões contextuais (ambos definidos usando sequências de tags gramaticais) para aprender a extrair padrões para cada predicado de interesse e usar esses padrões para encontrar instâncias adicionais de cada predicado. Relacionamentos entre predicados são usados para filtrar padrões que são muito gerais. Esse componente faz uso de aprendizado semissupervisionado.

3.2.6.3 CSEAL - Coupled SEAL

CSEAL é implementado com base na metodologia SEAL (Set Expander for Any Language)(WANG; COHEN, 2009). SEAL é um sistema de expansão que aceita elementos de entrada (sementes) de algum conjunto alvo *S* e automaticamente encontra outros prováveis elementos de *S* em documentos semiestruturados como páginas web ao consultar a web usando essas sementes.

No entanto, SEAL não possui um mecanismo para explorar as restrições de exclusão mútua e checagem de tipos, que são adicionadas no CSEAL. De um modo geral, CSEAL é um extrator de dados semiestruturados que realiza buscas na Internet com um conjunto de crenças para cada categoria ou relação, e então minera listas e tabelas para extrair novas instâncias do predicado

correspondente.

3.2.6.4 CMC - *Coupled Morphological Classifier*

CMC é um conjunto binário que classifica frases nominais baseando-se em várias características morfológicas. As crenças da base de conhecimento são utilizadas como instâncias de treinamento, mas a cada iteração o CMC está restrito a predicados que possuam ao menos 100 instâncias promovidas. Assim como o CSEAL, relacionamentos de exclusão mútua são utilizados para identificar instâncias negativas. CMC faz a análise de fatos candidatos propostos por outros componentes, e classifica até 30 novas crenças por candidato a cada iteração.

3.2.6.5 RL - *Rule Learner*

Rule Learner é um algoritmo de aprendizado relacional de primeira ordem semelhante ao FOIL (QUINLAN; CAMERON-JONES, 1993), o qual aprende cláusulas probabilísticas de Horn. Essas regras aprendidas são utilizadas para inferir novas instâncias de relação a partir das que já estão armazenadas na base de conhecimento.

3.2.6.6 Prophet

O Prophet é um componente de *link prediction* (LIBEN-NOWELL; KLEINBERG, 2007) que é acoplado ao NELL com o intuito de automaticamente auxiliar o sistema e expandir sua BC prevendo novos fatos e novas relações com alto grau de acurácia. *Link prediction* pode ser definido como ”Dada uma foto instantânea de um grafo G em um momento t, prever com precisão quais arestas irão aparecer em G no momento $t + 1$”(LIBEN-NOWELL; KLEINBERG, 2007).

A BC do NELL é utilizada como entrada. O Prophet tenta inferir fatos com base nos padrões presentes na base de conhecimento. Também pode criar novas relações (diferentemente do RL).

Considerando-se a sentença ”Milwaukee Bucks é um time de basquete que joga na liga NBA.”, tem-se que o NELL é capaz de extrair duas relações: EsporteTime(Basquete, Milwaukee Bucks) e timeJogaLiga(Milwaukee Bucks, NBA). Por outro lado, é natural o conhecimento humano inferir que ”Basquete é o esporte jogado na liga NBA”, o que não é tão simples para o NELL. Logo, o NELL precisa ter um componente que infere fatos em cima dos dados já presentes na BC. Assim, o objetivo do Prophet é estender o conceito tradicional de *link prediction* para ser aplicado em dados de rede complexos que representam conhecimento extraído da Web e, então, inferir novas relações e novos fatos que são representados pelas arestas.

3.2.6.7 Conversing Learning

Bases de conhecimento como a do NELL crescem continuamente, o que requere uma autossupervisão, bem como a autorreflexão dos dados obtidos. O crescimento da Internet proporcionou o aumento de mídias sociais. Dessa forma, o Conversing Learning (CL) propõe fazer uso do conhecimento coletivo obtido a partir de usuários da comunidade web, com o intuito de fornecer autossupervisão e autorreflexão para sistemas inteligentes para, então, poder aprimorar a tarefa de aprendizado.

A principal ideia do CL é procurar por fontes em comunidades web para confirmar alguma afirmação do NELL, caso os componentes do NELL não consigam chegar a uma conclusão. O CL busca automaticamente por supervisão humana na Web utilizando o Yahoo[6] e Twitter[7] com o intuito de aprimorar os métodos de aprendizado.

Um sistema como o CL deve ser capaz de responder às seguintes questões: (i) qual conhecimento deve ser disponibilizado aos humanos? (ii) quem são os humanos para quem a máquina deve pedir ajuda? (iii) Como entender as respostas dos humanos? (iv) Como inferir conhecimento a partir da resposta dos humanos?

Para explorar as capacidades do CL, é utilizado o algoritmo SS-Crowd (PEDRO; HRUSCHKA JUNIOR, 2012a), que utiliza a BC do NELL para unir uma máquina que aprende como um ser humano, e uma máquina que resolve suas questões como os humanos. O algoritmo SS-Crowd pode ser resumido pelas seguintes tarefas: (i) obter fatos da BC, (ii) construir uma pergunta entendível a partir dos fatos, (iii) consultar a comunidade da web com as perguntas, (iv) reunir e resolver as perguntas (classificá-las como positivas ou negativas), e (v) combinar as respostas e produzir uma opinião combinada da comunidade sobre a persistência dos fatos.

3.2.6.8 OntExt - Ontology Extension System

O*OntExt* propõe um método para descobrir relações relevantes automaticamente, dado um grande corpo textual e uma ontologia inicial definindo centenas de categorias nominais (como atletas, músicos e instrumentos). O OntExt descobre relações entre pares dessas categorias, utilizando dois passos. Primeiro, para cada par de categoria (por exemplo, músico e instrumento), ele coagrupa os contextos de textos que conectam instâncias conhecidas de duas categorias, gerando uma relação candidata para cada grupo resultante. Depois, ele aplica um classificador treinado para determinar quais dessas relações candidatas são semanticamente válidas.

[6] https://answers.yahoo.com/
[7] https://twitter.com/

A ideia principal é descobrir novas relações de forma automática, para estender a ontologia de sistemas como o NELL. O OntExt utiliza como dados de entrada (i) uma ontologia especificando um conjunto de categorias, (ii) uma BC contendo instâncias dessas categorias (talvez com erros), e (iii) um grande corpo textual. Como resultado, espera-se obter: (i) um conjunto de relações de dois argumentos que são frequentemente mencionados no texto, cujos tipos de argumento correspondem às categorias na ontologia de entrada (por exemplo, RioPassaPorCidade(<Rio>, <Cidade>)), (ii) para cada relação proposta, um conjunto de instâncias (como RioPassaPor Cidade(<Nilo>, <Cairo>)), e (iii) para cada relação proposta, um conjunto de padrões de extração de texto que podem ser utilizados para extrair instâncias adicionais da relação (por exemplo, o texto "X no coração de Y", em que X é um rio conhecido, e Y uma cidade conhecida, sugere extrair RioPassaPorCidade(X, Y)).

Para extrair relações de forma automática, OntExt possui três fases:

1. Iniciar explorando uma vasta página de texto na web;
2. Utilizar categorias e relações extraídas pelo CPL para gerar novas relações;
3. Após a geração das relações, um componente classificador para classificar semanticamente as relações válidas.

3.3 Técnicas envolvendo Grandes Bases de Conhecimento

Há várias técnicas distintas utilizadas para aumentar as grandes bases de conhecimento com fatos ou relações. Existem duas abordagens principais para serem utilizadas por algoritmos que visam à expansão de BCs: abordagem *interna* e *externa*. A primeira utiliza dos dados já armazenados nas BCs para descobrir informações ausentes, enquanto a segunda utiliza textos da Web para descobrir novos fatos e relações e expandir suas BCs (PAULHEIM, 2017). Algumas dessas técnicas já foram descritas em alguns dos componentes do NELL.

Em Gardner et al. (2013), os autores aprimoram o algoritmo PRA (Path Ranking Algorithm) (LAO; COHEN, 2010) aumentando o grafo da base de conhecimento adicionando bordas com rótulos sintáticos léxicos mais expressivos (em que rótulos são palavras em vez de dependências). O algoritmo PRA faz inferência direta por meio da base de conhecimento representada como grafo. Por exemplo, caso a BC possuísse os fatos, *(Tiger Woods, participaEm, PGA Tour)* e *(Golf, esporteDoCampeonato, PGA Tour)*, poder-se-ia inferir, colocando esses dois fatos juntos, que *(Tiger Woods, jogaEsporte, Golf)*.

PROSPERA (PRospering knOwledge with Scalability, PrEcision, e RecAll) (NAKASHOLE; THEOBALD; WEIKUM, 2011) é um algoritmo que tem o objetivo de popular a base de conhecimento com fatos, contendo alta precisão de qualidade. Ele propõe uma solução baseada em *n-gram-itemsets* para encontrar padrões semanticamente mais poderosos, e usa a restrição de raciocínio Max-Sat para obter a qualidade nos padrões e validar os fatos candidatos.

Uma técnica diferente e inovadora foi proposta por Vannella et al. (2014) para validar e estender uma base de conhecimento em larga escala. Eles criaram dois vídeo games para validar a relação entre conceitos ou entre conceito e imagem. Em comparação com o método de *crowdsourcing* (SARASUA; SIMPERL; NOY, 2012), as anotações obtidas, constantemente, possuíam maior qualidade.

GalÁrraga et al. (2013) criaram AMIE, um algoritmo de mineração de regras de associação aplicado em uma base com evidências incompletas. AMIE tem o foco de descobrir relações como *se mãeDe(m; c)* $\wedge$ *casadoCom(m; f)* $\rightarrow$ *paiDe (f; c)*. AMIE aplica o algoritmo em cima da BC já existente e possui os objetivos de aumentar a base (deixando-a menos incompleta) e encontrar suas inconsistências.

De acordo com Neelakantan e Chang (2015), a maioria dos trabalhos voltados para completar uma base de conhecimento (Knowledge Base Completion - KBC) focaram no problema de extração de relações. Para auxiliar a completar uma BC, eles se baseiam na tarefa de inferir instâncias de tipos de entidades ausentes. Eles construíram um grande conjunto de dados e projetaram uma metodologia de avaliação automática. Para popular a BC, eles utilizam informações já armazenadas na BC e informações externas da Wikipedia. Também desenvolveram um classificador para avaliar as técnicas para inferir instâncias de tipos de entidades ausentes.

DeepDive (SHIN et al., 2015) é um mecanismo de código aberto (*open-source*) para construção de base de conhecimento. É um sistema que combina ideias de banco de dados e aprendizado de máquina para auxiliar a desenvolver sistemas para a construção de BCs. Segundo os autores, a construção de BCs é um processo iterativo. Dessa forma, eles desenvolveram técnicas para produzir incrementalmente resultados de inferências para construir BCs. DeepDive utiliza uma linguagem baseada em *SQL* (do ponto de vista de banco de dados), e uma linguagem baseada na *Lógica de Markov* (DOMINGOS; LOWD, 2009) (do ponto de vista de aprendizado de máquina).

A maioria dos métodos atuais para popular bases de conhecimento não fazem uso de informações visuais (como imagens) que poderiam ser úteis para construir ou popular tais ontologias estruturadas (LI et al., 2016). Assim, Li et al. (2016) desenvolveram uma nova abordagem de mineração de padrões multimodais para a construção semiautomática de esquemas de evento

em alto nível. Eles utilizam um grande corpus de pares de legendas de imagens com pouca supervisão relacionados com eventos de alto nível como ”ataque”e ”demonstração”com o intuito de descobrir aspectos visuais de um evento, e de nomear esses componentes visuais automaticamente.

3.4 Grandes Bases de Conhecimento Inconsistentes

Um considerável problema na construção de grandes bases de conhecimento crescente é a quantidade de inconsistências que elas podem possuir. Existem várias razões que podem explicar a presença de incertezas nas BCs. De acordo com Benferhat, Dubois e Prade (1993), uma BC de conhecimento pode ser inconsistente pelos seguintes motivos:

- Presença de regras gerais com exceções;
- Existência de diversas fontes com possíveis discordâncias alimentando a BC.

Existem dois meios de tratar de conhecimento inconsistente (BENFERHAT; DUBOIS; PRADE, 1993):

- Revisar a BC e restaurar a inconsistência;
- Lidar com a inconsistência.

A primeira abordagem encontra duas dificuldades: (i) há várias maneiras de restaurar a inconsistência produzindo resultados diferentes, e (ii) parte da informação pode ser descartada e não será mais possível ter acesso a ela. A segunda abordagem resolve essas dificuldades.

Além disso, algoritmos desenvolvidos para construir grandes BCs a partir da web podem contribuir para trazer incertezas a uma BC. Segundo Dylla, Miliaraki e Theobald (2013a), métodos de extração de informação utilizados para estender uma grande BC podem obter fatos temporais incertos.

Vários trabalhos foram desenvolvidos visando à correção de inconsistências. Por exemplo, uma grande de variedade de técnicas para limpeza de dados, que são baseadas em *restrição de integridade* (BOHANNON et al., 2005; CHIANG; MILLER, 2011), estatística (MAYFIELD; NEVILLE; PRABHAKAR, 2010), ou aprendizado de máquina (YAKOUT; BERTI-ÉQUILLE; ELMAGARMID, 2013), foram propostas no passado. No entanto, elas não garantem a acurácia dos dados reparados, uma vez que não possuem evidência suficiente para identificar e atualizar os erros de

Tabela 3.1: Exemplo de tabela de jogadores de futebol

Tupla	A	B	C	D	E	F	G
t_1	Rossi	Itália	Roma	Verona	Italiano	Proto	1.78
t_2	Klate	África do Sul	Pretoria	Pirates	Sul-africano	P. Eliz	1.69
t_3	Pirlo	Itália	Madri	Juventus	Italiano	Flero	1.77

forma precisa (CHU et al., 2015). Por exemplo, considere-se a tabela 3.1 e a dependência funcional $B \rightarrow C$, que significa que B(país) determina unicamente C(capital). Poderia ser identificado um problema para os valores da tupla t_1 e t_3 para os atributos B e C. Um algoritmo de reparação de dados deveria adivinhar qual valor a ser trocado para limpar o dado.

Dessa forma, para tentar aumentar a acurácia de tais métodos, outros métodos como o uso de informação externa em tabelas de dados (FAN et al., 2010) e de especialistas do domínio (FAN et al., 2010; RAMAN; HELLERSTEIN, 2001) foram desenvolvidos. Todavia, esses recursos podem estar escassos ou terem alto custo de implantação.

Nos últimos anos, houve o crescimento de trabalhos envolvendo grandes bases de conhecimento, com várias abordagens sendo desenvolvidas para lidar com inconsistências. Do mesmo modo que os métodos para expandir as BCs, os métodos para detectar inconsistências ou erros se baseiam em abordagens internas ou externas. Os algoritmos que utilizam a abordagem interna usam como entrada os fatos e as relações já armazenados na BC. Por outro lado, a abordagem externa tenta descobrir erros em textos da Web, por exemplo. Os erros podem estar relacionados às asserções de tipo, às relações entre os indivíduos, ou entre os valores literais de uma categoria.

Dylla, Miliaraki e Theobald (2013a) propõem um modelo temporal probabilístico de banco de dados para limpar e corrigir as incertezas. Eles consideram a combinação de regras de dedução temporal, as restrições de consistências temporais e a inferência probabilística baseada na semântica de possíveis palavras comuns. Basicamente, eles adicionam uma probabilidade para os fatos temporais extraídos a partir de métodos de extração de informação.

Knowledge Vault (DONG et al., 2014b) é outro sistema que utiliza técnicas para tentar corrigir inconsistências em BCs. Eles desenvolveram um novo modo automático para a construção de uma BC a partir da Web. Para isso, ele combina extrações de ruídos da Web junto com o conhecimento previamente obtido. Por exemplo, supondo que um extrator retorne um fato dizendo que *Barack Obama* nasceu no Quênia, tem-se que o modelo prévio pode usar fatos relacionados sobre o *Obama* (como sua profissão de presidente dos Estados Unidos) para inferir que esse novo fato é improvável de ser verdadeiro.

KATARA (CHU et al., 2015) é um sistema que faz a limpeza de dados e bases de conheci-

mento. Ele recebe como dados de entrada uma BC e uma tabela. A partir desse ponto, *KATARA* interpreta a semântica de uma tabela para a alinhá-la à BC, identificando dados corretos e incorretos, e gera os *top-k* possíveis reparos para dados incorretos. O sistema pode ser utilizado em diversos conjuntos de dados e BCs, sugerindo de forma eficiente possíveis reparos.

Em Paulheim e Bizer (2014), foi proposto um método estatístico para encontrar informações erradas na base de conhecimento. Para cada tipo de relação, é calculada a distribuição de características do sujeito e dos tipos de objetos para a aresta do grafo, isto é, cada instanciação da relação. Arestas no grafo, cujo sujeito e tipo de objeto se desviam fortemente da característica de distribuição, são identificados como erros potenciais. O algoritmo usa o conhecimento interno da BC, ou seja, dados e relações já armazenados nas BCs.

Ngomo, Sherif e Lyko (2014) apresentaram o sistema *COLIBRI*, uma abordagem iterativa não supervisionada para a descoberta de links (*link discovery*). *COLIBRI* permite a descoberta de links entre várias BCs ao mesmo tempo que aprimora os dados nessas BCs. Para isso, ele combina detecção e correção de erros com descoberta de links não supervisionados.

Um método para identificar erros envolvendo tipos de asserções foi proposto por (MA et al., 2014). Eles utilizaram programação de lógica indutiva para aprender axiomas disjuntos e, então, aplicar esses axiomas para identificar potenciais tipos de asserções com erros.

Fan, Zhou e Zheng (2015) desenvolveram o algoritmo *IIKA* (Imperfect and Incomplete Knowledge Embedding), o qual é um modelo probabilístico que mede cada tripla ¡*h, r, t*¿ com o objetivo de identificar o melhor vetor de representação de baixa dimensão, minimizando a perda de confiança correspondente dada por métodos de aprendizado de máquina (como feitos pelo NELL) ou *crowdsourcing* (no caso do Freebase (BOLLACKER et al., 2008b)). Na tripla, *h* representa uma entidade "cabeça"(*head_entity*), *r* a relação (*relation*) e t a entidade "calda"(*tail_entity*).

Outros trabalhos utilizados para reparação de grandes bases de conhecimento podem ser encontrados em Hellmann et al. (2014), Cruz-Filipe et al. (2015), Lehmann e Bühmann (2010).

3.5 Considerações Finais

Conforme descrito nesta seção, grandes bases de conhecimento estão sendo cada vez mais usadas nos últimos anos. A maioria das técnicas que esses sistemas utilizam possuem a finalidade de expandir as bases de conhecimento utilizando os fatos e relações já existentes ou utilizando textos na Web. Além disso, muitos dos métodos utilizados para expandir tais bases

de conhecimento podem aprender dados com erros ou incertos. Portanto, o presente trabalho faz uso de extração de Regras de Associação e correlação temporal para popular a base de dados do NELL com instâncias, contribuindo para diminuir a sua incompletude, e utiliza métricas probabilísticas (a partir das STARs) e correlação temporal para detectar incertezas na base de conhecimento. Assim como alguns trabalhos citados, ambas técnicas são empregadas com dados já extraídos por outros componentes da NELL, isto é, com dados já existentes na sua BC.

Capítulo 4

VALORES AUSENTES

4.1 Definição

Dados reais frequentemente possuem imperfeições: podem conter erros, serem incompletos, possuírem incertezas ou serem vagos. Uma das formas de incompletude são os atributos com valores ausentes (*missing attribute values*) (GRZYMALA-BUSSE; HU, 2001).

Várias abordagens já foram implementadas para trabalhar com dados ausentes:

1. **Valor mais comum do atributo (Most Common Attribute Value)**. É considerado um dos métodos mais simples para lidar com valores ausentes. Aqui, o valor do atributo que ocorre com maior frequência é selecionado para ser o valor de todos os valores desconhecidos do atributo em questão.

2. **Valor de conceito mais comum do atributo (Concept Most common value attribute)**. Aqui são levadas em consideração as relações entre os atributos e uma decisão. É uma restrição do método anterior, em que o valor mais comum que ocorre com o conceito em questão é selecionado.

3. **C4.5**. Método baseado em entropia e na divisão do exemplo contendo valores ausentes de um atributo para todos os conceitos.

4. **Todos valores possíveis para o atributo**. O valor ausente é substituído por todos os valores conhecidos do atributo. Caso tenha mais de um valor ausente, o processo substitui o primeiro por um valor conhecido, o segundo por outro valor conhecido e, assim por diante.

5. **Ignorar exemplos com valores desconhecidos de atributos**. É o método mais simples,

em que são ignorados as instâncias que possuem pelo menos um valor desconhecido, e o resto da tabela é utilizado como entrada para o processo de aprendizagem.

6. **Tratar valores ausentes de um atributo como valores especiais**. Aqui, o valor desconhecido é tratado com uma abordagem totalmente diferente. Em vez de tentar encontrar algum valor conhecido para ele, o próprio valor desconhecido é considerado como um novo valor para o atributo que contém o valor ausente e ele é tratado da mesma forma que os outros valores.

4.2 Regras de Associação para Valores Ausentes

Algoritmos como o *Apriori* foram desenvolvidos com base em bancos de dados transacionais, em que não existe o problema de valores ausentes, uma vez que cada instância representa uma transação na tabela, sem considerar quais itens pertencem a determinados atributos. No entanto, eles são frequentes em bancos de dados relacionais (NAYAK; COOK, 2001). Nesses, cada dado em uma instância pertence a um atributo (coluna) na tabela. Dessa forma, um dado ausente representa um atributo que ainda não foi preenchido ou descoberto. De acordo com Ragel et al. (1998), existem algumas razões que explicam a ocorrência de dados ausentes em um conjunto de dados:

1. Valores armazenados são ausentes porque são muito pequenos ou grandes para serem mensurados;

2. Valores ausentes ocorrem pois podem ter sido esquecidos, perdidos ou não estavam disponíveis.

Existem algumas estratégias para trabalhar com o problema de valores ausentes em algoritmos de regras de associação. Em Nayak e Cook (2001), o algoritmo AR foi desenvolvido, para extrair regras de associação em bases de dados contendo ruídos ou valores ausentes. O AR é baseado no *Apriori* e utiliza dois processos principais para tratar dos valores ausentes e dos dados com ruídos. Primeiro, os valores ausentes são substituídos por uma distribuição probabilística sobre possíveis valores representados por dados existentes (técnica semelhante a de **todos valores possíveis para o atributo**, descrita na subseção anterior). Segundo, todos os dados contribuem probabilisticamente para os padrões de candidatos. Os padrões que recebem uma quantidade suficiente de suporte total ou parcial são mantidos e expandidos.

XMiner (CALDERS; GOETHALS; MAMPAEY, 2007) é outro algoritmo de regras de associação que trata o problema de bases de dados com valores ausentes. O algoritmo é baseado no trabalho

de Ragel et al. (1998), em que as noções de suporte e confiança são redefinidas, para lidar mais adequadamente com valores ausentes. O grau de suporte não é mais medido contra todo o banco de dados. Em vez, o suporte é definido com respeito a um subconjunto do banco de dados sem valores ausentes nos atributos do itemset. No entanto, eles aprimoram a eficiência do processo de extração de regras utilizando uma abordagem probabilística para ausência de dados utilizada em Kryszkiewicz (1999), em que as estimativas pessimistas, otimistas e esperadas de suporte e confiança são definidas.

Chen e Fan (2012) e Hong e Wu (2011) apresentam trabalhos com ênfase no problema de valores ausentes para mineração de regras de associação. Em Chen e Fan (2012), uma cadeia de Markov com base no método de Estimação de Valores Ausentes (*Missing Value Estimation (MC-MVE)*) foi proposta. *MC-MVE* adota uma abordagem iterativa para estimar o estado de probabilidade dos valores ausentes, com a probabilidade de transição do estado derivada de uma distribuição de Bernoulli, até atingir os critérios de convergência. Hong e Wu (2011) desenvolveram um método iterativo para extrair regras de associação inferindo valores ausentes com base no algoritmo criado em Nayak e Cook (2001).

4.3 Valores Ausentes em Grandes Bases de Conhecimento

Em grandes bases de conhecimento como NELL, YAGO, DBpedia e Cyc, valores ausentes estão presentes por razões diferentes das citadas por Grzymala-Busse e Hu (2001). De acordo com Miani, Pedro e Hruschka Junior (2014), existem dois principais motivos que justificam a presença de valores ausentes, além dos descritos por Grzymala-Busse e Hu (2001):

1. Valores ausentes ocorrem em grandes bases de conhecimento por não estarem disponíveis no momento da iteração de algum algoritmo;

2. Valores são ausentes pois determinada relação entre duas categorias pode nunca ocorrer.

Devido ao fato de grandes BCs obterem mais conhecimento a cada dia, essas bases são consideradas incompletas, principalmente pelos itens citados acima. Dessa forma, tem-se BCs com muitos valores ausentes. No caso específico do NELL, sistema em que este projeto está inserido, os algoritmos existentes extraem relações entre categorias como *atletaJogaEsporte (X,Y)* e *atletaJogaLiga (X, Z)*. Logo, pode-se induzir que, se um atleta X pratica um esporte Y, e, se o mesmo atleta X joga o torneio Z, então, esta competição (Z) está relacionada com o aquele esporte (Y). NELL pode também extrair relações como *atletaJogaEsporte (Lebron_James, basquete)*, *atletaJogaLiga (kevin_garnett, nba)* e *atletaJogaLiga (Lebron_James, nba)*. Note que,

Tabela 4.1: Exemplo de Valores Ausentes que podem nunca ocorrer

Atleta	Esporte	Torneio	Troféu
Lebron_James	basquete	nba	campeonato_nba
Roger_Federer	tênis	us_open	campeonato_us_open
Messi	Futebol	Liga_Campeões	campeonato_liga_campeões
Nene_hilario	basquete	nba	mv
Thomaz_Bellucci	tênis	us_open	mv

para esse exemplo, NELL não obteve a relação entre o atleta Kevin Garnett e o esporte que ele pratica e, portanto, esse valor será ausente na BC (Kevin Garnett também é jogador de basquete). Esse é o primeiro caso de valores ausentes descrito acima.

O outro caso de valor ausente em grandes bases de conhecimento diz respeito a quando a relação entre duas determinadas categorias ainda não ocorreu e pode nunca acontecer. Considere-se o exemplo amostral da tabela 4.1. Valores ausentes no pequeno conjunto de dados são ilustrados por *mv (missing value)*. Pode-se observar que há dois valores ausentes na tabela. Os dois casos são exemplos de valores que podem nunca ocorrer entre determinadas categorias. Ou seja, o atleta *Nene Hilario*, que joga *basquete* no campeonato *nba*, pode nunca ganhar o troféu de campeão dessa liga. Assim, o valor entre essas categorias pode nunca ser preenchido, sendo um caso de valor ausente. O mesmo se aplica para o tenista *Thomaz_Bellucci*, que pode nunca ganhar o título no torneio *us_open*.

4.4 Considerações Finais

Neste trabalho, é utilizado um algoritmo de regras de associação para auxiliar a popular a base de conhecimento do NELL com fatos. Uma vez que a BC do NELL é uma base incompleta, como mencionado anteriormente, ela possui muitos valores ausentes. Logo, o algoritmo de mineração de regras de associação escolhido (NARFO) foi modificado para tratar os valores ausentes. Para isso, o cálculo do suporte dos itemsets foi modificado, gerando um parâmetro adaptado denominado MSC (Modified Support Calculation). Para realizar este cálculo, são utilizadas duas aproximações das abordagens descritas (descarte de itemset e consideração do valor ausente como um atributo comum). O capítulo Arquitetura do Sistema traz em detalhes como é realizado esse cálculo, como o algoritmo descobre regras de associação para popular a base e expandir as relações entre as categorias do NELL.

Capítulo 5

TEMPORALIDADE

5.1 Definição

Pesquisas com bancos de dados temporais tiveram um crescimento explosivo nas décadas de 1980 e 1990 (MITSA, 2010). No entanto, a maioria dessas pesquisas focaram em sistemas de bancos de dados comerciais. Não existe uma linguagem bem aceita para a consulta de dados temporais, resultando no desenvolvimento de diferentes sistemas para extrair tais informações de um banco de dados tradicional. Um outro tópico emergente é a descoberta de temporalidade em informações semânticas, que utilizam o contexto de ontologias temporais.

Mitsa (2010) cita três tipos de entidades temporais que podem ser armazenadas em um banco de dados:

- *Intervalo (interval)*: uma entidade temporal com tempos iniciais e finais;
- *Evento (event)*: uma entidade temporal associada a uma ocorrência de tempo;
- *Séries Temporais (time series)*: consiste em uma série de medidas de valores reais em intervalos regulares.

Outras medidas importantes a serem consideradas em bancos de dados temporais são:

- *Granularidade*: descreve a duração da medida de tempo. Por exemplo, a granularidade pode ser semana ou dia;
- *Dados Ancorados (Anchored data)*: o tempo é representado usando valores absolutos, como 20 de Janeiro, 1999, 3:15 am;

- *Dados não Ancorados (Unachored data)*: são utilizados para representar a duração de um intervalo, como 2 semanas;

- *Coalescência de Dados (Data coalescing)*: a substituição de duas tuplas A e B por uma única tupla C, em que A e B possuem atributos não temporais idênticos, e intervalos temporais adjacentes ou sobrepostos. C tem os mesmos atributos não temporais, enquanto seu intervalo temporal é a união dos intervalos de A e B.

Koubarakis (1990) provê uma lista de requisitos que devem ser preenchidos por um sistema de gerenciamento de bases de conhecimento temporal:

- Para ser possível responder consultas do mundo real, o sistema deve ser capaz de tratar grandes quantidades de dados temporais;

- Deve ser capaz de responder a consultas sobre relações temporais quantitativas e qualitativas;

- Deve ser capaz de representar causalidade entre dados temporais;

- Deve ser capaz de diferenciar entre a história de um evento e o tempo que o sistema aprendeu sobre o evento.

De acordo com Mitsa (2010), a mineração tradicional de dados temporais foca principalmente na informação temporal: relacionamentos de *causa/efeito* por meio da descoberta de regras de associação, classificação, agrupamento, entre outras. Como mencionado anteriormente, nos últimos anos, cresceu outra abordagem de mineração de dados temporais que engloba a descoberta de relacionamentos estruturais e semânticos, em que estes são realizados com o conceito de ontologias temporais.

Além da mineração de dados temporais, a temporalidade tem ganhado bastante importância em sistemas que trabalham com grandes bases de conhecimento. Em seus primórdios, esses sistemas focavam apenas em extrair da web fatos e possíveis relações entre esses fatos. Nos últimos anos, identificar o tempo em que determinada relação ocorreu é um assunto de grande importância e que pode trazer conhecimento com mais precisão. Hoffart et al. (2013), Talukdar, Wijaya e Mitchell (2012) e Dylla, Miliaraki e Theobald (2013b) são alguns trabalhos que lidam com o problema de dados temporais em GBCs.

O *Intervalo Temporal Algébrico de Allen (Allen's TIA - Temporal Interval Algebra)*(ALLEN, 1983) é um formalismo bem conhecido para trabalhar com conhecimento temporal. Os sistemas

Tabela 5.1: Relações Temporais de Allen

Relação Temporal de Allen	Condições para ambos intervalos	Representação
meets(A,B)	a- $<$ b-, a- $<$ b+ a+ = b-, a+ $<$ b+	AAAA BBBBBBB
before(A,B)	a- $<$ b-, a- $<$ b+ a+ $<$ b-, a+ $<$ b+	AAAA BBBBBBB
overlaps(A,B)	a- $<$ b-, a- $<$ b+ a+ $>$ b-, a+ $<$ b+	AAAA BBBBBBB
starts(A,B)	a- $=$ b-, a- $<$ b+ a+ $>$ b-, a+ $<$ b+	AAAA BBBBBBB
finishes(A,B)	a- $>$ b-, a- $<$ b+ a+ $>$ b-, a+ $=$ b+	AAAA BBBBBBB
during(A,B)	a- $>$ b-, a- $<$ b+ a+ $>$ b-, a+ $<$ b+	AAAA BBBBBBB
equal(A,B)	a- $=$ b-, a- $<$ b+ a+ $>$ b-, a+ $=$ b+	AAAAA BBBBB

computacionais baseados no intervalo de Allen geralmente representam um período de tempo T como um intervalo de dois pontos temporais T = [t-, t+], que satisfaz a condição t- $<$ t+, em que t- e t+ são, respectivamente, as datas do evento inicial e final.

Assumindo que períodos temporais são definidos por pontos temporais, os intervalos A = [a-, a+] e B = [b-, b+] são representações válidas de períodos temporais se a- $<$ a+ e b- $<$ b+, em que a-, a+, b-, b+ $\subseteq \mathbb{R}$. Com base em dois períodos temporais específicos A e B, o intervalo algébrico de Allen define sete possíveis correlações temporais entre eles, o que é mostrado na tabela 5.1. Na terceira coluna, tem-se a representação das relações. O conjunto pode ser expandido para 13 se as relações reversas são consideradas (note que o reverso da relação *equal* é igual a ela mesma). De acordo com Nicoletti et al. (2012), dependendo do domínio da aplicação, os pontos finais de um intervalo representando períodos temporais dificilmente podem ser definidos com precisão e, normalmente, há um grau de incerteza sobre seus valores. Nesse sentido, os autores modificaram o intervalo algébrico convencional de Allen ($<$t-, t+$>$) para representar incertezas temporais. Eles definem uma quádrupla ($<$t–, t-, t+, t++$>$), em que t–, t-, t+, t++ $\subseteq \mathbb{R}$ and t– $<$ t- $<$ t+ $<$ t++.

Na próxima seção são descritas algumas técnicas de mineração de dados temporais e de abordagens que utilizam temporalidade em GBCs.

5.2 Técnicas com dados Temporais

Nesta seção, são abordadas técnicas de mineração de dados temporais, bem como algoritmos e ferramentas que envolvem temporalidade em grandes bases de conhecimento.

5.2.1 Mineração de Dados Temporais

O objetivo da mineração de dados temporais é descobrir relações escondidas entre sequências e subsequências de eventos. Dependendo da natureza do evento sequencial, as abordagens para resolver o problema podem ser diferentes. Uma sequência composta por uma séria de símbolos nominais a partir de um alfabeto particular é usualmente chamada de *sequência temporal* e uma sequência de valores reais e contínuos é denominada *série temporal* (ANTUNES; OLIVEIRA, 2001). Em geral, as aplicações para sequências temporais servem principalmente para realizar diagnósticos e prever comportamentos futuros.

Sequências e séries temporais aparecem naturalmente em uma variedade de domínios, desde engenharia até pesquisa científica, finanças e medicina. Em problemas de engenharia, eles normalmente surgem com monitoramento baseado em sensores, como controle de telecomunicações (Das, Gunopulos e Mannila (1997), Das et al. (1998)), ou em monitoramento de sistemas baseados em logs (Mannila e Toivonen (1996), Mannila e Meek (2000)). Na pesquisa científica, sequências e séries temporais podem aparecer em missões espacias (Keogh e Smyth (1997), Oates (1999)) e em domínios genéticos (Das et al. (1998)).

Na área financeira, aplicações envolvendo análise de vendas de produtos ou inventário de consumidores são de grande importância para o planejamento de negócios (Agrawal e Srikant (1995)). Outra aplicação muito comum em finanças é a predição da evolução de dados financeiros (como os trabalhos feitos em Faloutsos, Ranganathan e Manolopoulos (1994), Chan e Fu (1999)).

Muitos trabalhos foram desenvolvidos na área da saúde utilizando mineração de dados temporais. Em Berlingerio et al. (2007), uma análise da mineração de dados temporais foi utilizada em cima de dados de pacientes que realizaram transplantes de fígado. Szvarça et al. (2016) utilizaram regras de associação temporal em câncer de mama.

A mineração temporal de dados, assim como a tradicional, possui várias tarefas para realizar o processo, como a descoberta de *Regras de Associação*, *Classificação* e *Agrupamento* não supervisionado. Nas subseções a seguir, são apresentados alguns trabalhos de mineração de dados temporais utilizando essas tarefas.

5.2.1.1 Regras de Associação

O problema da descoberta de regras de associação (AGRAWAL; IMIELINSKI; SWAMI, 1993) já foi tratado no capítulo 2. Aqui são descritos alguns trabalhos e técnicas que envolvem a mineração de regras de associação envolvendo dados temporais.

Uma extensão do o algoritmo *Apriori* para trabalhar com mineração sequencial foi proposta por Agrawal e Srikant (1995). O processo de descoberta de sequências temporais é similar ao do *Apriori*, em que o suporte de uma sequência é definido como a fração de tuplas em que a sequência está contida. O algoritmo GSP (SRIKANT; AGRAWAL, 1996) foi proposto pela mesma equipe que desenvolveu o *Apriori* para trabalhar com sequências de dados temporais, mas é cerca de 20 vezes mais rápido que o *Apriori*.

Uma abordagem trazida por Das et al. (1998) consiste em estender a tradicional regra $X \Rightarrow Y$ para uma regra com um novo significado: $X \Rightarrow^T Y$, significando que se X ocorrer, Y também está presente com relação ao tempo T. Com isso, pode ser possível controlar o impacto da ocorrência de um evento em outra ocorrência, com um específico intervalo de tempo.

Em Ozden, Ramaswamy e Silberschatz (1998), tem-se um método que consiste em encontrar regras cíclicas. Uma *regra é cíclica* se ocorre em intervalos regulares de tempo, isto é, transações que suportam específicas regras que ocorrem periodicamente (por exemplo, "*toda primeira segunda do mês*"). Para descobrir tais regras, é necessário procurá-las em porções restritas de tempo, uma vez que podem ocorrer repetitivamente em instâncias específicas de tempo mas em uma pequena porção do tempo total considerado. Pode-se descobrir as regras a partir de um algoritmo similar ao *Apriori*, e procurar por ciclos de regras escondidos nas mesmas. Uma abordagem mais eficiente consiste em descobrir os ciclos nos itemsets e, depois, gerar as regras. Uma extensão para esse método considera a existência de diferentes unidades de tempo, como dias, semanas ou meses, e é realizada definindo um calendário algébrico para definir e manipular grupos de intervalos de tempo. As regras descobertas nesse processo são denominadas de *Calendric Association Rules* (Regras de Associação Calendric) (RAMASWAMY; MAHAJAN; SILBERSCHATZ, 1998).

Em Winarko e Roddick (2007), o algoritmo ARMADA foi desenvolvido para descobrir regras de associação temporais semanticamente mais ricas a partir de dados baseados em intervalos. Eles também adicionaram uma restrição, denominada *maximum gap*, que pode ser usada para remover padrões insignificantes.

Outros trabalhos que envolvem mineração de regras de associação temporais podem ser encontrados em Ale e Rossi (2000), Talha e Junejo (2014), Shaheen, Shahbaz e Guergachi (2013) e Cariñena (2014).

5.2.1.2 Classificação

Segundo Antunes e Oliveira (2001), a tarefa de classificação é uma das mais utilizadas em aprendizado supervisionado. No entanto, seu uso em mineração temporal de dados ganhou maior atenção a partir do final dos anos de 1990.

Um método de classificação que trabalha com sequências temporais é baseado no operador *merge* (KEOGH; PAZZANI, 1998). Este recebe duas sequências e retorna uma sequência cuja forma é um ajuste entre as duas sequências originais. A ideia básica é realizar o *merge* iterativamente a partir de um exemplo típico de uma classe contendo cada exemplo positivo, construindo um modelo mais geral para a classe. Isso é alcançado com a utilização de um fator de influência para controlar a função do operador *merge*.

Antunes e Oliveira (2001) também relata, que algoritmos tradicionais de classificação são difíceis de aplicar em exemplos sequenciais, principalmente porque existe um vasto número de características potencialmente úteis para descrever cada exemplo. Para tentar contornar tal questão, Lesh, Zaki e Ogihara (1999) implementam um mecanismo de pré-processamento para extrair características relevantes. Esta abordagem consiste na descoberta de subsequências frequentes e, então, as utiliza como características relevantes para classificar sequências com métodos tradicionais, como *Naive Bayes* ou *Winnow*.

Amaral et al. (2014) utiliza um método de classificação semissupervisionada de séries temporais de índices de vegetação obtidas a partir de *SITS (Satellite Image Time Series)*, visando à identificação de áreas de plantio de cana-de-açúcar. Tseng e Lee (2009) descreveram um novo método de mineração de dados, denominado *CBS* (classify by sequence), para classificar grandes conjuntos de dados temporais. A metodologia principal por trás do *CBS* é integrar a mineração de padrões sequenciais com indução probabilística.

Em Lian et al. (2016) é utilizada uma estratégia de sugestões visuais estáticas e espaciotemporais para isolar regiões de interesses (*ROIs*) estáticas e espaciotemporais. Eles utilizam o aprendizado fracamente supervisionado para treinar classificadores de redes profundas usando os *ROIs* como entrada. Mais precisamente, eles combinam aprendizado de múltiplas instâncias (MIL - multiple instance learning) com redes neurais convolutivas (CNNs - Convolutional neural networks) para selecionar sugestões de ações discriminativas. Isso produz classificadores para imagens estáticas, usando os ROIs estáticos, bem como classificadores para sequências de imagens curtas (16 quadros), minerando ROIs espaciotemporais.

Em Chen, Ye e Hu (2007), um algoritmo para classificação de séries temporais foi proposto, com base na discretização de séries temporais usando uma *medida de entropia*. O objetivo do

processo de discretização é dividir as séries temporais em k intervalos conforme um critério de divisão. O critério de divisão é a minimização da entropia dentro de cada intervalo. A entropia de um intervalo é 0 (zero), se todos os valores do intervalo pertencerem à mesma classe. No entanto, se todas as classes aparecerem iguais frequentemente em um intervalo, então a entropia do intervalo alcança seu máximo. Seguindo a discretização da entropia, a série temporal é transformada em séries de strings. Depois, o método de classificação *k-nearest neighbor (k-vizinhos mais próximos)* é utilizado.

Outros trabalhos que utilizam classificação para a mineração de dados temporais podem ser encontrados em Patel, Hsu e Lee (2008), Fleischman, Decamp e Roy (2006), Moskovitch e Shahar (2014), Batal et al. (2012) e Fradkin e Mörchen (2015).

5.2.1.3 Agrupamento

Agrupar sequências ou séries temporais está relacionado a agrupar uma coleção de séries temporais (ou sequências) com base na sua similaridade (LAXMAN; SASTRY, 2006). O agrupamento é de interesse particular em mineração de dados temporais, uma vez que ele fornece um mecanismo atrativo para se encontrar automaticamente uma estrutura em grandes conjuntos de dados que seriam difíceis de visualizar.

O principal problema em realizar o agrupamento de sequências temporais em bancos de dados consiste em descobrir o número K de grupos (clusters) capazes de representar as diferentes sequências (ANTUNES; OLIVEIRA, 2001). Existem dois problemas centrais em agrupamento: (i) a escolha do número de grupos e inicialização dos seus parâmetros, e (ii) a existência de medidas significativas de similaridades entre as sequências.

Considere-se que K é conhecido. Se a sequência é vista como sendo gerada de acordo com algum método probabilístico, como o modelo de Markov, o agrupamento pode ser visto modelando as sequências de dados como um grupo finito de K sequências na forma de um modelo finito misto. Por meio do algoritmo EM, seus parâmetros podem ser estimados e cada grupo K deve corresponder a um grupo (SMYTH, 1999). O valor de K, se desconhecido, pode ser encontrado pelo método de validação cruzada de Monte-Carlo, como sugerido em Smyth et al. (1997).

Existem vários trabalhos que utilizaram agrupamento para mineração de dados temporais. Um método de agrupamento hierárquico para agrupar sequências temporais foi proposto em Ketterlin (1997). O algoritmo usado foi o COBWEB (FISHER, 1987), e ele trabalha em dois passos: (i) agrupar os elementos das sequências, e (ii) agrupar as sequências por elas mesmas.

Em Fujita et al. (2012) foi desenvolvida uma técnica para agrupamento de genes através da identificação da causalidade de Granger Granger (1969) entre e dentro de conjuntos de dados de expressão gênica de séries temporais. A causalidade de Granger é baseada na ideia de que a causa de um evento não pode ocorrer após a sua consequência.

Sefidmazgi et al. (2014) utilizam agrupamento de séries temporais climáticas não estacionárias para encontrar pontos de mudanças com múltiplas tendências lineares. Para isso eles mostram como o método de agrupamento baseado no método de elemento finito (FEM - finite element method) pode ser utilizado para localizar pontos de mudanças na tendência das séries temporais de temperatura a partir de observações *in situ* (no local).

Alon et al. (2003) desenvolveram um método de agrupamento de séries temporais de dados baseado em HMMs (Hidden Markov Models). O método é aplicado às séries temporais de dados resultantes de movimentos de objetos, como os de humanos, animais, e veículos. Dois métodos iterativos foram utilizados para estimar os parâmetros da função de probabilidade: o k-means e o algoritmo EM.

Birant e Kut (2007), Qamra, Tseng e Chang (2006), Lee (2012) e Zhang e Eick (2014) apresentam outros exemplos de métodos que trazem o agrupamento não supervisionado no processo de descoberta de relações implícitas na base de dados.

5.2.2 Temporalidade em Grandes Bases de Conhecimento

Pesquisas envolvendo grandes bases de conhecimento ganharam grande foco nos últimos anos. A maioria dos sistemas tem o intuito de expandir a BC, populando-a com novos fatos ou relações. No entanto, a preocupação com o tempo exato em que um determinado fato ou uma relação ocorreu é um campo ainda pouco explorado, tendo obtido maiores esforços a partir de 2010. Relações temporais podem tornar uma sentença verdadeira ou falsa. Dessa forma, vários trabalhos foram desenvolvidos envolvendo a temporalidade nas relações para expandir a base de conhecimento crescente.

YAGO2 (HOFFART et al., 2013), o qual estende a base de conhecimento do YAGO, considera o tempo e espaço no armazenamento de fatos, entidades e eventos, com acurácia de 95%.

Kuzey e Weikum (2012) relatam que a maioria das BCs atuais possuem esforços apenas em fatos estáticos, ignorando a dimensão temporal. No entanto, a maioria dos fatos que envolvem tempo são válidos apenas em determinado período. Em seu trabalho, eles construíram um framework completo para extração de informação que procura por fatos e eventos temporais de dados semiestruturados e de texto de artigos do Wikipedia para criar uma ontologia temporal.

O sistema possui precisão de 90% para dados semiestruturados e de 70% para dados temporais extraídos de textos. Eles estenderam a representação do conhecimento da ontologia T-YAGO (WANG et al., 2010) para fatos e eventos temporais, criando medidas estatísticas para ranquear os padrões extraídos.

Com o intuito de responder a questões como *Quais países George W.Bush visitou em 2006?* ou *Quantas vezes o presidente Bush encontrou o primeiro ministro Brown em 2007?*, Wang et al. (2011b) desenvolveu um método para a coleta de dados envolvendo tempo e espaço a partir de artigos de notícias.

PRAVDA (WANG et al., 2011a) é um sistema que une fatos candidatos e refina os fatos com sua extensão temporal com base em uma nova forma de *label propagation* (propagação de rótulos). Essa é uma família de métodos de aprendizado semissupervisionado baseados em grafos. Inicialmente, eles possuem um pequeno número de fatos corretamente rotulados.

CoTS (TALUKDAR; WIJAYA; MITCHELL, 2012) é um framework de inferência para escopo temporal, que potencializa restrições de fatos temporais específicos com fraca supervisão, tendo poucos rótulos de exemplos. Apenas poucos fatos rotulados são necessários para treinar um classificador local. Ele tenta verificar se um fato específico como *Bill Clinton-presidenteDo-US* é válido durante o período de 1993-2001.

Lacasta et al. (2014),Hautala e Jauhiainen (2014) e Ji et al. (2014) são outros trabalhos que desenvolveram algoritmos para tratar de dados temporais na construção de suas bases de conhecimento. No primeiro, é descrito um processo que facilita a criação de uma BC contendo a evolução do domínio jurídico espanhol entre 1830 e 2011. Hautala e Jauhiainen (2014) apresentam um novo framework para a criação de uma base de conhecimento envolvendo espaço e tempo. Eles avaliaram quatro grupos de pesquisas na Finlândia nos campos de tecnologia e ciências. Com a combinação de espaço e tempo foi possível obter diferentes aspectos de conhecimento. Por fim, Ji et al. (2014) apresentam um trabalho contendo três principais perspectivas: (i) representação da informação temporal, em que as teorias e limitações na área são vistas e um novo framework é desenvolvido para a tarefa; (ii) aquisição de anotação, ao propor um método de regressão logística para aprimorar a qualidade dos rótulos dos dados treinados por supervisão distante; (iii) classificação da informação temporal, em que duas abordagens são desenvolvidas e combinadas por meio da agregação de documentos cruzados.

5.3 Considerações Finais

Este capítulo descreveu conceitos de temporalidade em bases de conhecimento, além de técnicas de mineração de dados temporais e como a temporalidade é tratada em grandes BCs. Com o intuito de auxiliar no processo de popular a base de conhecimento do NELL, este projeto fez uso de regras de associação temporal específicas (STARs) e da correlação temporal entre as STARs e as demais instâncias da BC, visando tanto a popular a base de conhecimento (para diminuir a quantidade de valores ausentes) como a detectar possíveis inconsistências na BC. A mineração de dados temporais é tratada de forma diferente das técnicas abordadas nesta seção. Primeiro, é realizada a mineração de regras de associação. Depois, a partir de cada regra gerada, obtêm-se as regras de associação temporais específicas, que foram utilizadas para gerar a regra. Esse processo é detalhado no próximo capítulo. As STARs são utilizadas para (i) detectar possíveis inconsistências encontradas pelos componentes TARE e TCI e (ii) auxiliar a descobrir correlações temporais para popular a BC.

A BC do NELL possui poucos fatos com dados temporais. Dessa forma, para fazer uso de dados temporais em uma grande BC, alguns dados temporais foram rotulados, da mesma forma como se fazia com a base do NELL em seus primórdios.

Capítulo 6

ARQUITETURA DO SISTEMA

6.1 Introdução

Este capítulo descreve as técnicas e métodos utilizados para auxiliar a expansão da base de conhecimento do NELL neste trabalho de doutorado, bem como os procedimentos realizados para detectar possíveis incertezas presentes na base. Para estender a BC do NELL, faz-se uso de (i) extração de regras de associação e (ii) correlação temporal (componente TCI). Ambos têm o intuito de expandir a BC, populando-a com instâncias. Para realizar a detecção de incertezas, são utilizadas as métricas probabilísticas com base nas regras de associação temporais específicas extraídas (componente TARE) e o módulo de detecção de inconsistências do componente TCI.

Como previamente mencionado, o NELL é um sistema que possui uma base de conhecimento com crescimento contínuo e, dessa forma, não possui milhares de fatos e relações do mundo. Ao se aplicar um algoritmo de regras de associação em uma BC com essas características, deve-se levar em consideração a presença de valores ausentes. Dessa forma, o cálculo do suporte dos itemsets frequentes é modificado para tratar desse problema. Esse procedimento é realizado durante a geração de candidatos e esse novo cálculo é denominado de MSC (Modified Support Calculation). O MSC produz itemsets com suportes maiores, o que resulta em uma maior quantidade de regras geradas (com mais regras úteis para serem utilizadas para povoar a BC), sendo esta uma das contribuições deste trabalho.

Um problema frequente em algoritmos com base em itemsets frequentes é a grande quantidade de regras geradas. Analisar cada uma pode ser uma tarefa árdua. Assim, o presente trabalho apresenta o componente ER, que propõe algumas formas para de diminuição do esforço tanto ao analisar as regras de associação quanto as regras temporais específicas. Para as regras

de associação, dois procedimentos são realizados no pós-processamento das regras:

1. Eliminação de regras de associação redundantes;
2. Eliminação de regras de associação irrelevantes.

O primeiro item consiste em eliminar as denominadas *super-regras antecedentes*. No entanto, a regra só é eliminada do conjunto de regras de associação extraídas se o algoritmo encontra todas as possíveis sub-regras com os antecedentes da super-regra a ser eliminada. Esse procedimento é descrito na subseção Eliminar Regras Redundantes. Para o segundo item, são eliminadas as super-regras consequentes consideradas irrelevantes com base em uma sub-regra irrelevante que possua o(s) mesmo(s) antecedente(s), além de sub consequente irrelevante em comum. Esse processo é melhor explicado na seção Eliminar Regras Irrelevantes. A introdução das super-regras antecedentes e consequentes é primeiramente descrita neste trabalho, assim como a técnica para eliminação desses tipos de regras, sendo contribuições deste projeto.

Outro ponto explorado neste trabalho de doutorado é a temporalidade em grandes bases de conhecimento. Como já mencionado no capítulo 5, poucas BCs consideram o tempo quando um determinado evento ocorreu. A BC do NELL possui poucos dados com informações temporais. Dessa forma, dados temporais foram rotulados para alguns fatos da atual BC do NELL. Com isso, é possível descobrir o intervalo de tempo em que o fato ocorreu. A temporalidade é explorada em dois componentes. O componente TARE mostra todas as regras de associação temporais específicas (STARs), cujo intervalo temporal correspondes às datas inicial e final em que o evento ocorreu. As regras específicas são todas as instâncias utilizadas para gerar uma regra de associação. Isso será detalhado na seção que descreve o componente TARE. Cada STAR é utilizada como entrada para o componente TCI para (i) encontrar correlações temporais para estender a BC, e (ii) detectar possíveis inconsistências com base em discordâncias entre as STARs e as demais instâncias da base.

Com o intuito de detectar inconsistências na base de dados, ambos os componentes TARE e TCI possuem contribuições. O primeiro utiliza duas métricas probabilísticas (MP1 e MP2), que verificam a probabilidade de uma regra ser pontual ou não. Logo, as regras que não se enquadram na porcentagem indicada são separadas para verificar se são inconsistentes. Já o componente TCI utiliza a correlação temporal entre uma STAR e as instâncias da base de dados para verificar possíveis imprecisões. Para realizar essas duas funcionalidades do TCI, o intervalo temporal de Allen (TIA) é modificado para se adequar ao procedimento utilizado, reduzindo de 13 para 11 o número de intervalos temporais.

Este trabalho possui dois objetivos principais: (i) popular a base de conhecimento do NELL e (ii) auxiliar no processo de detecção de incertezas presentes na mesma. Para isso, várias técnicas foram desenvolvidas, tendo as seguintes contribuições:

1. Uso de regras de associação para auxiliar a popular uma grande base de conhecimento;
2. Criação do parâmetro MSC para algoritmos de regras de associação em uma grande base de conhecimento crescente contendo muitos valores ausentes;
3. A investigação sobre o quão efetivo e necessário é analisar cada regra de associação descoberta;
4. Apresentação das *super-regras antecedentes* e de um novo processo de eliminação das consideradas redundantes;
5. Apresentação das *super-regras consequentes* e de um novo processo de eliminação das consideradas irrelevantes;
6. Introdução de regras de associação temporais específicas;
7. Avaliação das regras de associação temporais específicas extraídas por meio de métricas probabilísticas para detectar inconsistências na base de dados;
8. Adaptação do intervalo algébrico temporal de Allen para realizar a atualização e a detecção de inconsistências na base de dados;
9. Atualizar a BC com base na correlação temporal;
10. Detectar inconsistências com base na correlação temporal.

A figura 6.1 mostra a arquitetura do sistema. Os retângulos destacados são as etapas em que novas técnicas são introduzidas. Note que, deles, saem flechas que levam às principais inovações desse trabalho. É interessante observar que é extraído um subconjunto de dados do NELL para ser utilizado pelo algoritmo. Logo, o algoritmo de regras de associação utiliza esse subconjunto para gerar as regras. As regras consideradas relevantes são utilizadas para expandir a base de conhecimento do NELL. Com base nas regras extraídas, o componente TARE obtém as regras de associação temporais específicas que originaram cada regra de associação. As STARs que são consideradas inconsistentes pelas métricas probabilísticas são avaliadas para possíveis correções. O restante é utilizado pelo componente TCI para obter correlações temporais que possam povoar a base de conhecimento e detectar inconsistências. Após a correção das

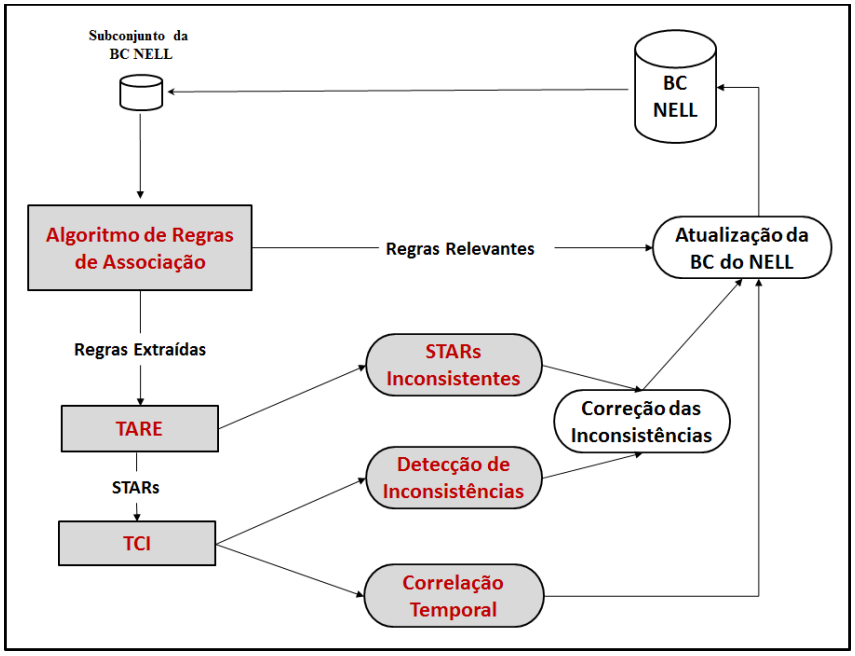

Figura 6.1: Arquitetura do Sistema

inconsistências, a base também é atualizada com os fatos corrigidos. Esse processo se repete a cada iteração do algoritmo, uma vez que o ambiente utilizado é de aprendizado sem fim. As partes destacadas com fundo preenchido são as que possuem contribuições.

A seção 6.2 descreve o algoritmo de regras de associação utilizado, o parâmetro MSC, o componente ER, que traz os métodos para eliminar regras redundantes e irrelevantes, e informa como avaliar regras de associação em grandes bases de conhecimento incompletas. As STARs e a descrição das métricas probabilísticas utilizadas são detalhadas na seção 6.3. Por fim, a seção 6.4 mostra o componente TCI e explica como a correlação temporal atua no processo de detecção de inconsistências e de extensão da base de conhecimento.

6.2 Algoritmo de Regras de Associação

Para realizar a extração de regras de associação, foi escolhido como algoritmo base o NARFO (Non-redundant and generalized Association Rules based on Fuzzy Ontologies) (MIANI et al., 2009). Ele foi escolhido por, além de minerar regras de associação, também extrair regras generalizadas, sendo capaz de navegar pela estrutura ontológica que representa a base de dados do NELL. Também é capaz de verificar a relação entre ancestral e descendente, que, para a base do NELL, corresponde a suas categorias (ancestrais) e fatos (descendentes). Por exemplo, *esporte* seria uma categoria e *tênis* seria um fato associado a ela. Além dos tradicionais parâmetros de suporte e confiança, o NARFO traz duas outras medidas (que não são utilizadas

neste trabalho):

- *minsim* (ESCOVAR; YAGUINUMA; BIAJIZ, 2006), que considera o grau de similaridade entre dois ou mais itens da base de dados;
- *minGen (minimal Generalization)* (MIANI et al., 2010), que considera uma porcentagem mínima para generalizar itens de um domínio específico.

Assim como nas medidas de suporte e confiança das regras, os valores para essas medidas são definidos por um valor real no intervalo [0, 1].

No entanto, algumas modificações foram feitas em sua implementação para realizar os objetivos e contribuições deste trabalho. Para um algoritmo de regras de associação trabalhar em uma base incompleta como a do NELL, foi criado o parâmetro MSC para tratar de valores ausentes em grandes BCs. Além disso, para diminuir o esforço ao analisar as regras extraídas, foi desenvolvido o componente ER contendo dois métodos: (i) eliminação de regras de associação redundantes, e (ii) eliminação de regras de associação irrelevantes. Por fim, antes de utilizar as regras obtidas para popular a BC do NELL, cada regra extraída deve ser analisada em virtude da característica da BC, que não possui todos os valores para cada categoria.

A figura 6.2 ilustra as etapas do algoritmo para a extração de regras de associação. As partes com fundo destacado são as que possuem contribuições. Um subconjunto da BC do NELL é extraído e utilizado como entrada para o algoritmo. Após a preparação dos dados, tem-se a etapa de geração dos candidatos, em que é realizado o cálculo do suporte considerando valores ausentes por meio do parâmetro MSC. A próxima fase do processo é a geração de regras a partir dos itemsets frequentes. Além da geração das regras, a eliminação de regras redundantes e irrelevantes é realizada para diminuir o esforço ao avaliá-las. Por fim, com auxílio do Conversing Learning, cada regra é avaliada, e as regras relevantes são utilizadas para popular a BC do NELL.

6.2.1 Preparação dos Dados

Nesta etapa é realizada a preparação da base de dados a ser utilizada pelo algoritmo. Um subconjunto dos dados do NELL é extraído. Com base nele, uma ontologia, contendo suas categorias e fatos, é desenvolvida no Protégé (KNUBLAUCH et al., 2004). Como já mencionado, a base de dados do NELL é composta por valores ausentes. Para indicar tal item na base de dados, o valor *mv* foi incluído em cada célula que não contém valor para determinada categoria.

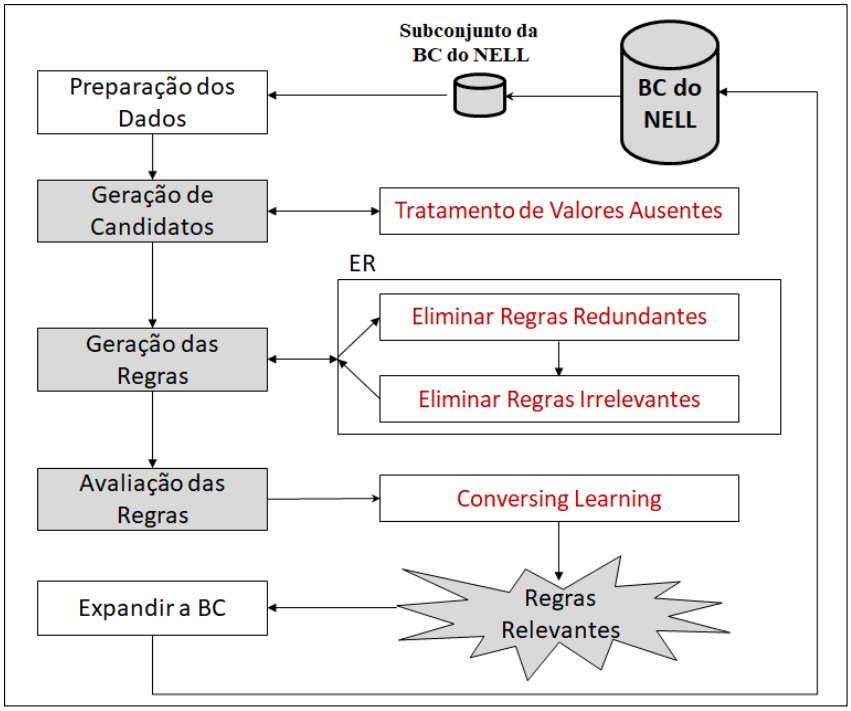

Figura 6.2: Etapas do Algoritmo de Regras de Associação

6.2.2 Geração dos Candidatos

Este passo é similar ao *Apriori* (AGRAWAL; IMIELINSKI; SWAMI, 1993), tendo o objetivo de obter itemsets frequentes. Inicialmente, o suporte de cada 1-itemset (itemset de tamanho 1) é calculado. Os itemsets de tamanho 1 que tiverem suporte maior ou igual ao mínimo desejado são considerados frequentes. Após este procedimento é feito o passo de junção (join). Nele, cada 1-itemset é unido com outros 1-itemset formando conjuntos de 2-itemset candidatos (itemsets de tamanho 2). E o processo continua até que não seja mais possível encontrar itemsets frequentes. Para cada n-itemset (itemset de tamanho n) é realizado o cálculo do seu respectivo suporte.

Contudo, para realizar o cálculo do suporte em bases de conhecimento contendo valores ausentes, foi desenvolvido um novo parâmetro, denominado MSC, que é uma contribuição deste trabalho.

6.2.2.1 Cálculo do suporte considerando valores ausentes

Uma grande base de conhecimento como a do NELL possui muitas células com valores ausentes. Para realizar um algoritmo de mineração de regras de associação em tal ambiente, é necessário tratar esses valores. Assim, o cálculo tradicional do suporte foi adaptado, criando um novo parâmetro, denominado MSC (Modified Support Calculation), para tratar os valores ausentes. Resumidamente, essa técnica descarta um itemset durante a contagem do suporte dos

Tabela 6.1: Conjunto amostral de dados.

Atleta	Esporte	LigaEsportiva	Troféu	TimeEsportivo
ben_roethlisberger	Futebol Americano	nfl	super_bowl	pittsburgh_steelers
brian_urlacher	Futebol Americano	nfl	mv	bears
favre	Futebol Americano	nfl	mv	jets
joe_flacco	mv	nfl	mv	mv
larry_foote	Futebol Americano	nfl	super_bowl	pittsburgh_steelers
steve_mcnair	Futebol Americano	nfl	mv	mv
tom_brady	Futebol Americano	nfl	super_bowl	new_england_patriots
drew_bledsoe	mv	mv	super_bowl	new_england_patriots

Algoritmo 4: Pseudo-algoritmo para cálculo do MSC

Entrada: *listaCandidatos*
Saída: *listaCandidatos*

para $i = 0$ *a* $listaCandidatos.tamanho() - 1$ **faça**
  $contaValorAusente = 0$;
  $itemset = listaCandidatos.getItemset(i)$;
  $peso = getPeso()$;
  $todos_itens = verificaLinhaDominio(itemset)$;
  se $todos_itens == verdadeiro$ **então**
    $contaValorAusente = contaValorAusente + 1$;
  fim
  $suporte = peso \div (numeroLinhas - contaValorAusente)$;
  $listaCandidatos.add(itemset, suporte)$;
fim

itemsets se todos seus itens são ausentes.

Considere-se o exemplo amostral representado na tabela 6.1, o qual contém instâncias de 4 categorias distintas (*Esporte, LigaEsportiva, Troféu e TimeEsportivo*) relacionadas à categoria *atleta*. Células com o valor *mv* na tabela 6.1 representam valores ausentes. Logo, ao calcular o suporte de um itemset usando a medida MSC, o algoritmo não conta os valores ausentes quando todos os itens dos respectivos domínios estão ausentes. Na tabela 6.1, a categoria *Esporte* possui duas células vazias, e elas não são contabilizadas durante o cálculo do suporte para 1-itemset. As categorias *LigaEsportiva*, *Troféu* e *TimeEsportivo* possuem, respectivamente, uma, quatro e duas células vazias. A tabela 6.2 apresenta os valores de suporte dos itemsets utilizando o parâmetro MSC. O pseudo-algoritmo desse procedimento é mostrado no Algoritmo 4.

Em cada fase da geração de itemsets frequentes, o algoritmo analisa cada itemset candidato. Para cada um, verificam-se os domínios do itemset atual e, caso exista uma tupla com esses domínios contendo todos os valores ausentes, ela não é considerada para o cálculo do suporte. Ou seja, a medida MSC computa da seguinte forma: para um *n-itemset*, o algoritmo descarta um valor ausente do cálculo do suporte se *n itens* das categorias dos valores do itemset corrente não estiverem presentes. Por exemplo, em um conjunto de *1-itemset*, se no domínio desse item

Tabela 6.2: Comparativo entre FIMV x FI x FCI x MFI

Itemset	Suporte pelo MSC	Suporte Tradicional	FIMV	FI	FCI	MFI
Futebol Americano	6/6 = 1	6/8 = 0.75	X	X		
nfl	7/7 = 1	7/8 = 0.87	X	X	X	
super_bowl	4/4 = 1	4/8 = 0.5	X	X	X	
pittsburgh_steelers	2/6 = 0.33	2/8 = 0.25	X			
football, nfl	6/7 = 0.85	6/8 = 0.75	X	X	X	
nfl, super_bowl	3/8 = 0.37	3/8 = 0.37	X	X		
Futebol Americano, super_bowl	3/7 = 0.42	3/8 = 0.37	X	X		
pittsburgh_steelers, Futebol Americano	2/7 = 0.28	2/8 = 0.25				
Futebol Americano, nfl, super_bowl	3/8 = 0.37	3/8 = 0.37	X	X	X	X

no conjunto de dados, um valor não estiver presente, esse não é considerado para o cálculo do suporte. Para *2-itemset*, ambos os itens das categorias do itemset atual devem estar ausentes para que seja descartado. O algoritmo continua até que nenhum itemset possa ser encontrado.

Para cada *itemset* encontrado, o algoritmo pega a quantidade de linhas em que todos os itens do atual itemset aparecem. Então, ele verifica o número de linhas contendo todos os valores ausentes nos domínios do presente itemset. Finalmente, o cálculo do suporte é realizado, a partir da divisão do números de ocorrências do itemset pelo número total de linhas menos as linhas com valores ausentes para o itemset corrente. Pela tabela 6.1, o domínio *TimeEsportivo* possui 6 (seis) valores em um total de 8 linhas. Logo, tem-se dois valores ausentes que são descartados na contagem do suporte. Os valores do suporte dos itens *pittsburgh_steelers*, *bears*, *new_england_patriots* e *jets* são, respectivamente, 2/6 (1/3 = 0.333), 1/6 (0.167), 2/6 (1/3 = 0.333) e 1/6 (0.167), uma vez que os valores ausentes não foram considerados para o total de linhas. Analise agora o 2-itemset *pittsburgh_steelers, Futebol Americano*. É possível notar que, no conjunto de dados, a quarta linha para os dois domínios (*TimeEsportivo* e *Esporte*) não possui valores. Assim, esta linha não é considerada durante o cálculo do suporte desse itemset.

O parâmetro de suporte MSC proporciona o aumento do valor do suporte comparado com o cálculo tradicional. Isso é resultado do descarte das células ausentes. A tabela 6.2 também traz o cálculo do suporte tradicional. Por exemplo, temos que o suporte de *Futebol Americano, super_bowl* é 3/7 (0.42). Sem o novo cálculo, o mesmo suporte seria de 3/8 (0.37). O aumento do valor do suporte faz com que mais regras sejam geradas, dependendo dos valores mínimos desejados, o que leva a padrões e regras que poderiam não ser descobertos por métodos tradicionais. Assim, essa nova medida traz ganho na qualidade das regras geradas, além de trazer novas regras que não seriam obtidas por métodos tradicionais de regras de associação.

Além disso, a tabela 6.2 também mostra um comparativo envolvendo quais itemsets seriam gerados pelo parâmetro MSC em comparação com algoritmos envolvendo cálculo do suporte

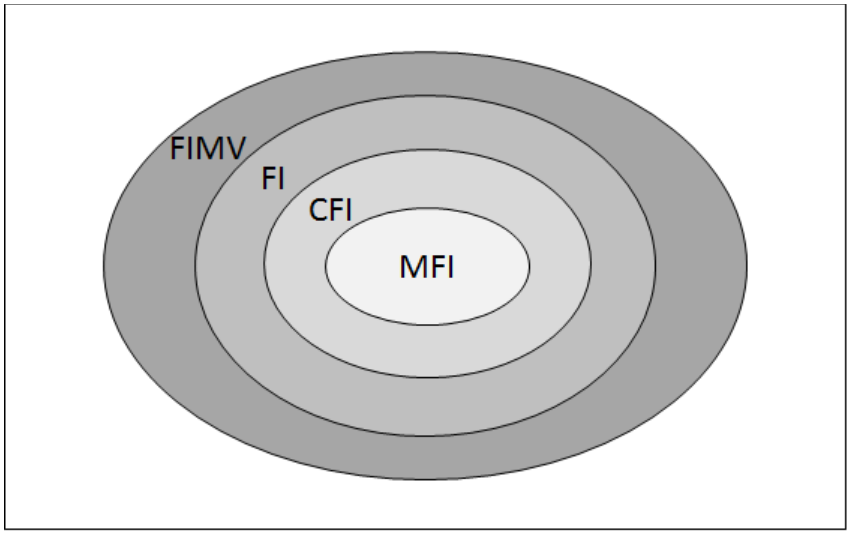

Figura 6.3: Relacionamento entre FIMV x FI x FCI x MFI

tradicional, com FCIs (Frequent Closed Itemsets - Itemsets Frequentes Fechados) e MFIs (Maximal Frenquent Itemsets), que foram abordados no capítulo 2. Considere-se a abordagem de itemsets frequentes com valores ausentes sendo FIMVs (Frequent Itemsets with Missing Values). Tem-se o seguinte relacionamento MFI $\subseteq$ FCI $\subseteq$ FI $\subseteq$ FIMV, conforme a figura 6.3.

Pela tabela 6.2, pode ser observado o relacionamento mostrado na figura 6.3: MFIs (1) $\subseteq$ FCIs (4) $\subseteq$ FIs (7) $\subseteq$ FIMVs (8). Ao utilizar os FIMVs, o algoritmo irá gerar mais regras (na próxima etapa) que podem auxiliar no processo de popular a BC do NELL. Se for considerado o conjunto de FCIs e MFIs, os itemsets (*Futebol Americano, super_bowl*) e (*nfl, super_bowl)* não serão gerados. Portanto, regras de associação como *AR*1 e *AR*2 não serão geradas, o que contribui para popular a BC com menos fatos em comparação com a técnica desenvolvida neste trabalho.

*AR*1: *atletaGanhaTrofeu(X, super_bowl)* $\rightarrow$ *atletaJogaEsporte(X, footbal),* e

*AR*2: *atletaGanhaTrofeu(X, super_bowl)* $\rightarrow$ *atletaJogaLiga(X, nfl).*

6.2.3 Geração das Regras de Associação

Semelhantemente ao que ocorre no *Apriori*, esta etapa gera as regras de associação com base no conjunto de itemsets frequentes. As regras que tiverem grau de suporte e confiança maior ou igual aos desejados são consideras regras fortes. Para cada itemset frequente, essa etapa obtém todas as combinações de antecedente/consequente que o itemset pode formar. Por exemplo, considere-se o itemset *Futebol Americano, nfl, super_bowl* da tabela 6.2. A tabela 6.3

Tabela 6.3: Exemplo de regras geradas para o itemset *(Futebol Americano, nfl, super_bowl)*

Regra	Suporte	Confiança
atletaJogaEsporte(X, Futebol Americano) → atletaJogaLiga(X, nfl), atletaGanhaTrofeu(X, super_bowl)	0.37	0.37
atletaJogaLiga(X, nfl) → atletaJogaEsporte(X, Futebol Americano), atletaGanhaTrofeu(X, super_bowl)	0.37	0.37
atletaGanhaTrofeu(X, super_bowl) → atletaJogaLiga(X, nfl), atletaJogaEsporte(X, Futebol Americano)	0.37	0.37
atletaJogaEsporte(X, Futebol Americano), atletaJogaLiga(X, nfl) → atletaGanhaTrofeu(X, super_bowl)	0.37	0.43
atletaJogaEsporte(X, Futebol Americano), atletaGanhaTrofeu(X, super_bowl) → atletaJogaLiga(X, nfl)	0.37	0.88
atletaJogaLiga(X, nfl), atletaGanhaTrofeu(X, super_bowl) → atletaJogaEsporte(X, Futebol Americano)	0.37	1

traz todas as possíveis regras que esse itemset pode gerar. Imagine que o suporte e confiança mínimos sejam 0.3 e 0.4, respectivamente. Apenas as três últimas regras da tabela 6.3 seriam mostradas no final do processo.

Além da geração de regras de associação, foi desenvolvido o componente ER (Eliminating Rules), o qual possui dois métodos para eliminação de regras redundantes e irrelevantes com o intuito de otimizar o resultado obtido e diminuir o esforço ao avaliá-las. Assim, este passo possui os seguintes objetivos, sendo os dois últimos contribuições deste trabalho:

1. Gerar regras de associação;
2. Eliminar regras de associação redundantes;
3. Eliminar regras de associação irrelevantes.

Na sequência são descritos os dois métodos do ER: (i) Eliminar Regras Redundantes, o qual consiste em retirar super-regras antecedentes consideradas redundantes, e (ii) Eliminar Regras Irrelevantes, que elimina super-regras consequentes irrelevantes.

6.2.3.1 Eliminar Regras Redundantes

Antes de explicar o procedimento de eliminação das regras redundantes e mostrar quais tipos de regras serão excluídas, algumas definições são necessárias. Considere $X \rightarrow Y$, $X' \rightarrow Y$ como regras de associação, em que X' é um superconjunto de X.

Definição 4 *Uma regra de associação $X' \rightarrow Y$ é uma Super-Regra Antecedente (super antecedent rule - super AR) se ela possui os mesmos itens no consequente e se seu antecedente X' é um superconjunto de um antecedente X. Por outro lado, uma regra de associação $X \rightarrow Y$ é uma Sub-Regra Antecedente (sub antecedente rule - sub AR) se existe outra regra com o mesmo consequente e se X é um subconjunto de X'.*

Tabela 6.4: Exemplo de regras de associação para a Tabela 6.2.

Número	Regra de Associação
1	*atletaGanhaTrofeu(X, super_bowl) → atletaJogaEsporte(X, Futebol Americano)*
2	*atletaJogaLiga(X, nfl) → atletaJogaEsporte(X, Futebol Americano)*
3	*atletaGanhaTrofeu(X, super_bowl), atletaJogaLiga(X, nfl) → atletaJogaEsporte(X, Futebol Americano)*
4	*atletaJogaEsporte(X, Futebol Americano) → atletaJogaLiga(X, nfl)*
5	*atletaJogaEsporte(X, Futebol Americano) → atletaJogaLiga(X, nfl), atletaJogaTime(X, pittsburgh_steelers)*
6	*atletaJogaEsporte(X, Futebol Americano), atletaJogaTime(X, pittsburgh_steelers) → atletaJogaLiga(X, nfl)*

Todas as *super-regras antecedentes* são *super-regras*, mas nem todas *super-regras* são *super-regras antecedentes*.

Considere-se a tabela 6.4 que traz possíveis regras de associação a partir dos itemsets da tabela 6.2. A regra 3 é uma *super-regra antecedente* das regras 1 e 2, e a regra 6 é uma *super-regra antecedente* da regra 4. No entanto, a regra 5 é uma *super-regra*, mas não uma *super AR*. Para ser uma *super AR*, a regra tem que ter mais de um item, pelo menos, no antecedente, o que não é o caso da regra 5.

O algoritmo de eliminação de regras redundantes consiste em remover *super-regras antecedentes*. Contudo, ele somente retira a regra do conjunto de regras de associação se ele encontra todas as suas possíveis *sub-regras antecedentes* da super AR. Pelo exemplo da tabela 6.4, a regra 3 pode ser eliminada uma vez que as regras 2 e 3 são todas as suas possíveis sub-regras antecedentes. No entanto, a regra 6 não pode ser retirada pois somente uma de suas possíveis sub ARs foi gerada. Para popular uma BC, uma sub AR é mais eficiente do que sua *super AR*, o que justifica a eliminação da super AR.

Os exemplos da tabela 6.4 representam apenas super ARs contendo dois itens no lado do antecedente. Dessa forma, o algoritmo procura por sub-regras antecedentes de 1-itemset no lado do antecedente. Porém, o que ocorre nos casos em que a super AR possua 3 itens (3-itemset) ou mais no seu antecedente? Para o caso de 3 itens no antecedente da super AR, devem existir sub ARs com 1 ou 2-itemsets no lado do antecedente, por exemplo. Para solucionar esse problema, o algoritmo calcula todas as possíveis combinações de sub ARs conforme a equação 6.1, a qual representa o cálculo matemático de uma combinação simples. O Algoritmo 5 mostra o pseudo-código desse método.

$$Combinação(n,k) = \frac{n!}{k!(n-k)!} \tag{6.1}$$

Algoritmo 5: Algoritmo de Eliminação de Regras Redundantes

Entrada: *listaDeRegras*
Saída: *listaDeRegrasSemRedundancia*

```
1 todasCombinações = falso;
2 para i = 0 a numeroDeRegras − 1 faça
3 |   regraAtual = getAssociationRule(i);
4 |   todasCombinações = encontreTodasComb(regraAtual);
5 |   se !(todasCombinações) então
6 |   |   listaDeRegrasSemRedundancia.add(regraAtual);
7 |   fim
8 |   todasCombinações = falso;
9 fim
```

Para uma melhor compreensão, considere a tabela 6.5 como exemplo. Na primeira linha, existe uma super-regra antecedente de 4 itens (4-itemset) no lado antecedente da regra. O algoritmo tem início tentando encontrar todas as possibilidades de sub-regras antecedentes com 1 item (1-itemset). Pela equação 6.1, n é o tamanho de itens no antecedente da super AR, e k é o tamanho itens no antecedente da sub AR. Logo, existem 4 possíveis sub ARs de 1-itemset no antecedente. Porém, somente 3 regras foram geradas, e a super AR de 4-itemset no antecedente ainda não pode ser eliminada. A seguir, o algoritmo continua a procurar por todas as possíveis combinações de sub ARs de tamanho 2 no lado do antecedente da regra. Para o exemplo da tabela 6.5, todas as 6 possibilidades foram geradas. Assim, a regra da primeira linha pode ser eliminada. Esse é um exemplo de super AR considerada redundante.

Se forem consideradas as regras com 3 itens no lado antecedente como super ARs, ambas são eliminadas. A regra $a,b,c \rightarrow e$ possui todas as possibilidades de sub ARs de tamanho 1 no antecedente, e $a,b,d \rightarrow e$ possui todas as possibilidades de tamanho 2. Para o primeiro caso, o algoritmo começa investigando as possibilidades de sub ARs de tamanho 1. Como encontra todas as possibilidades nessa etapa, o algoritmo termina o procedimento e remove a regra do conjunto final de regras. Já para a regra $a,b,d \rightarrow e$, o algoritmo só encontra todas as possibilidades de regras com suas sub ARs de tamanho 2, realizando uma iteração a mais.

A principal tarefa do algoritmo é encontrar todas as possibilidades sub ARs de tamanho 1 até (n-1) no lado antecedente, onde n é o tamanho do antecedente da super AR corrente (encontreTodasComb no Algoritmo 5). Caso todas as possíveis combinações forem encontradas com 1-itemset, o algoritmo retira a regra de associação do conjunto. Caso contrário, ele tenta encontrar todas as possibilidades de sub ARs de 2-itemsets, e assim por diante.

Se forem comparadas as regras geradas pelo método aqui proposto e por algoritmos de FCIs e MFIs, é mais provável que as super-regras sejam geradas nestes, enquanto aquele propõe

Tabela 6.5: Exemplo para eliminar regras super antecedentes

Tamanho do Antecedente	Regra de Associação	Número de combinações
4	a,b,c,d → e	-
3	a,b,c → e / a,b,d → e	4
2	a,b → e / a,c → e / a,d → e / b,c → e / b,d → e / c,d → e	6
1	a → e / b → e / c → e	4

eliminá-las, mantendo as sub ARs mínimas, que são mais eficazes para popular grandes BCs. Por exemplo, se algoritmos de FCIs MFIs descobrirem somente o itemset (*a,b,c,d,e*) como closed e máximo, ele só poderia popular a BC com o valor *e* se todos os outros itens aparecerem na mesma instância na base de dados. Contudo, ao utilizar a técnica de eliminação de regras redundantes descrita neste trabalho, a regra $a,b,c,d \rightarrow e$ é eliminada, resultando somente nas sub ARs de tamanho 2 no lado antecedente como descrito anteriormente. Dessa forma, quando uma das seis possibilidades de sub ARs estiver presente em uma instância, o algoritmo preenche o conjunto de dados com o valor de *e* quando necessário. Isso contribui para popular grandes BCs compostas por valores ausentes mais do que quando usadas regras de associação extraídas por algoritmos de FCIs e MFIs.

6.2.3.2 Eliminar Regras Irrelevantes

Após a eliminação das super ARs redundantes, é realizado o processo de verificar possíveis super-regras irrelevantes, que serão retiradas do conjunto final de regras. Em cada iteração, o algoritmo pode extrair uma quantidade de regras irrelevantes, principalmente em razão de a BC do NELL ser incompleta e crescer a cada dia. Ao final da primeira iteração do algoritmo (primeira vez que é executado), nenhuma regra de associação irrelevante é previamente conhecida. Portanto, o procedimento não elimina nenhuma regra irrelevante na primeira iteração, tendo maior eficácia a partir da segunda iteração do primeiro ciclo de iterações. Cada ciclo é executado com diferentes valores para o suporte mínimo desejado. Por exemplo, executar o algoritmo com suportes 0.2 e 0.1 com uma determinada base é considerado um ciclo. Após o final do primeiro ciclo, o subconjunto da BC do NELL é aumentado, e o próximo ciclo pode ser executado. Todas as regras irrelevantes descobertas em uma iteração *i* serão úteis para auxiliar a eliminar futuras regras de associação irrelevantes em iterações $i + n$, com $n \geq 1$, assim como em futuros ciclos de iteração.

Definição 5 *Uma regra de associação é considerada irrelevante se possuir a capacidade de popular uma grande base de conhecimento crescente com dados incorretos, contribuindo com a propagação de erros.*

Considere-se novamente a tabela 6.2 que traz valores de suporte para alguns itemsets. A tabela 6.4 traz algumas regras de associação geradas a partir da tabela 6.2. As regras 2 e 4 são relacionadas ao 2-itemset *Futebol Americano, nfl*. Agora, suponha que as regras 2 e 4 tenham sido geradas na primeira iteração, no ciclo inicial, que tinha o valor mínimo para o suporte em 0.3. Dessa forma, o itemset é frequente, e as regras 2 e 4 são geradas.

Todas novas regras extraídas são avaliadas pelo CL, componente do NELL. A regra 2 da tabela 6.4 é considerada relevante e será utilizada para popular a BC, substituindo os valores ausentes para a categoria *Esporte* com o valor de *Futebol Americano*, sempre que o valor *nfl* aparecer em uma instância para a categoria *Liga*. Por exemplo, a linha 4 da tabela 6.1 possui uma célula vazia relacionada à categoria *Esporte*, e o valor correspondente para a *Liga* é *nfl*. Logo, o algoritmo preenche essa célula com o valor de *Futebol Americano*. Contudo, a regra 4 é considera irrelevante, já que nem todos os atletas que praticam *Futebol Americano* jogam a liga esportiva *nfl*. Embora os dados amostrais da tabela 6.1 sugiram a regra como sendo forte, o CL a avalia como irrelevante uma vez que um atleta de *Futebol Americano* pode jogar em outras ligas. Nesse caso, a regra 4 é considerada irrelevante e não pode auxiliar a preencher a BC do NELL, uma vez que irá populá-la com informação incorreta.

Na segunda iteração, o valor do suporte mínimo desejado foi definido em 0.2. Logo, o 3-itemset (*Futebol Americano, nfl, super_bowl*) é considerado no processo de geração de regras. Suponha que o algoritmo tenha gerado a seguinte regra:

1. *atletaJogaEsporte(X, Futebol Americano)* $\rightarrow$ *atletaJogaLiga(X, nfl), atletaGanhaTrofeu(X, super_bowl).*

Essa regra de associação também é considerada irrelevante. No entanto, existe uma sub-regra (regra 4 da tabela 6.4) que também é irrelevante, e que foi descoberta em uma iteração anterior. Assim, essa regra é eliminada do conjunto, não sendo avaliada ao final do processo. É importante observar que a super-regra irrelevante eliminada deve ter o mesmo antecedente da sub-regra irrelevante. Portanto, o algoritmo elimina todas as super-regras que possuem o mesmo antecedente de uma sub regra irrelevante.

Uma super-regra contendo mais antecedentes não pode ser eliminada, uma vez que a adição de mais antecedentes a uma regra irrelevante, pode torná-la relevante. Considere-se a seguinte regra de associação:

2. *atletaJogaEsporte(X, Futebol Americano), atletaGanhaTrofeu(X, super_bowl)* $\rightarrow$ *atletaJo-*

Algoritmo 6: Algoritmo de Eliminação de Regras Irrelevantes

Entrada: *listaRegrasSemRedundancia*
Saída: *regrasFinais*

```
1 ehSuperRegraIrrel = falso;
2 para i = 0 a numeroDeRegrasNãoRedundantes − 1 faça
3 |   regraAtual = getAssociationRule(i);
4 |   ehSuperRegraIrrel = encontreSubIrrel(regraAtual);
5 |   se ) então !(hasSuperIrrelRule
6 |   |   regrasFinais.add(regraAtual);
7 |   fim
8 |   ehSuperRegraIrrel = falso;
9 fim
```

gaLiga(X, nfl).

Embora a regra 2 acima possua uma sub-regra que é irrelevante (regra 4 da tabela 6.4), a adição do itemset *super_bowl* (no lado do antecedente) transformou a regra em relevante. Neste caso, o troféu *super_bowl* está relacionado aos atletas que ganham a liga *nfl*.

Considere $X \rightarrow Y$, $X \rightarrow Y'$ como regras de associação, em que Y' é um super conjunto de Y.

Definição 6 *Uma regra de associação $X \rightarrow Y'$ é uma Super-Regra Consequente (super consequent rule - super CR) se ela possui os mesmos itens no antecedente e se seu consequente Y' é um superconjunto do consequente Y. Por outro lado, uma regra de associação $X \rightarrow Y$ é uma Sub-Regra Consequente (sub consequente rule - sub CR) se existe outra regra com o mesmo antecedente e Y é um subconjunto de Y'.*

Todas as *super-regras consequentes* de uma sub-regra consequente irrelevante também são irrelevantes. Todas *super-regras consequentes* são *super-regras*, mas nem todas *super-regras* são *super-regras consequentes*.

O Algoritmo 6 descreve o procedimento de eliminação de regras irrelevantes. O conjunto de regras não redundantes obtido pelo método de eliminação de regras redundantes é utilizado como entrada. Para cada regra gerada, o algoritmo verifica se ela possui uma sub CR irrelevante. Caso possua, a regra é retirada do conjunto final de regras e não é avaliada no final da iteração.

6.2.4 Avaliação das Regras Geradas

Algoritmos tradicionais de mineração de regras de associação consideram fortes os padrões extraídos que possuem grau de suporte e confiança maiores ou iguais aos mínimos desejados.

Tabela 6.6: Regras Finais.

Número	Regra de Associação
1	*atletaGanhaTrofeu(X, super_bowl) → atletaJogaEsporte(X, Futebol Americano)*
2	*atletaJogaLiga(X, nfl) → atletaJogaEsporte(X, Futebol Americano)*
3	*atletaJogaEsporte(X, Futebol Americano), atletaJogaTime(X, pittsburgh_steelers) → atletaJogaLiga(X, nfl)*

Entretanto, isso não se aplica a grandes bases de conhecimento crescentes, uma vez que a BC possui muitos valores ausentes e não contém todos os fatos para as categorias. Assim, uma regra considerada forte pode trazer um falso conhecimento, o que pode resultar na propagação de erros caso o mesmo seja utilizado para estender grandes BCs. Dessa forma, o processo de avaliação das regras de associação em ambientes de crescimento contínuo é necessário, sendo uma importante contribuição deste trabalho.

Para melhor exemplificar essa situação, observe a tabela 6.4, que possui regras geradas a partir da tabela 6.1. Os algoritmos do NELL não extraíram todas os fatos para todas as categorias, resultando em várias instâncias com valores ausentes. A tabela 6.6 traz as regras finais, retirando as super ARs redundantes e as irrelevantes. Pela tabela 6.4, como relatado anteriormente, as regras 4 e 5 (suporte 0.3 e 0.2 nos exemplos acima, respectivamente) são irrelevantes. Na iteração com suporte 0.2 a regra 5 sequer é ilustrada ao final do processo, uma vez que a regra 4 foi analisada para o suporte 0.3, e considerada irrelevante. Assim, a regra 4 não é utilizada para povoar a BC do NELL, uma vez que iria contribuir para a propagação de erros.

Para realizar a análise de cada regra resultante extraída na seção anterior, é utilizado um componente do NELL, denominado Conversing Learning. Esse, como descrito no capítulo 3, utiliza usuários do Twitter e do Yahoo Answers para validar as regras extraídas. Contudo, neste trabalho, apenas os usuários do Twitter foram utilizados. Com o auxílio do Conversing Learning, pretende-se avaliar automaticamente as regras extraídas e utilizar as regras validadas para aumentar a base de conhecimento do NELL.

6.2.5 Atualização da Base de Dados

Com as regras de associação analisadas pelo Conversing Learning, as positivas (validadas) podem ser utilizadas para atualizar e popular a base de conhecimento do NELL. Para cada regra de associação válida extraída, o algoritmo atualiza a BC do NELL com os valores do consequente da regra sempre que encontrar os antecedentes da mesma presentes em uma instância. Por exemplo, considerando a regra 1 da tabela 6.6, tem-se que o algoritmo irá atualizar todas as instâncias com o valor de *Futebol Americano* para a categoria *Esporte* sempre que a instância

Tabela 6.7: Conjunto amostral de dados atualizado com as regras de associação.

Atleta	Esporte	LigaEsportiva	Troféu	TimeEsportivo
ben_roethlisberger	Futebol Americano	nfl	super_bowl	pittsburgh_steelers
brian_urlacher	Futebol Americano	nfl	mv	bears
favre	Futebol Americano	nfl	mv	jets
joe_flacco	Futebol Americano	nfl	mv	mv
larry_foote	Futebol Americano	nfl	super_bowl	pittsburgh_steelers
steve_mcnair	Futebol Americano	nfl	mv	mv
tom_brady	Futebol Americano	nfl	super_bowl	new_england_patriots
drew_bledsoe	Futebol Americano	mv	super_bowl	new_england_patriots

possuir o valor de *super_bowl* para a categoria *Troféu*. Isso contribui para preencher a BC, diminuindo a quantidade de valores ausentes.

A tabela 6.7 traz a tabela 6.1 atualizada com a substituição dos valores ausentes utilizando as regras da tabela 6.6. Note que algumas células da categoria *Esporte* foram preenchidas com o valor *Futebol Americano*.

Esse é um dos procedimentos utilizados neste trabalho para auxiliar a popular a BC do NELL. O outro é realizado pelo componente TCI, que é descrito na seção 6.4

6.3 TARE - Temporal Association Rule Extraction

Este componente é executado imediatamente após a geração das regras de associação. Ele possui dois subcomponentes com os seguintes objetivos:

- Obter as regras de associação temporais específicas (STARs) que originaram cada regra gerada;
- Verificar possíveis incertezas na base de conhecimento por meio das métricas probabilísticas.

As STARs obtidas trazem o intervalo de tempo em que ela ocorreu. Ou seja, aqui é considerada a temporalidade para a BC do NELL. As STARs podem ser pontuais ou não pontuais. Com base na sua característica pontual, as métricas probabilísticas procuram descobrir possíveis inconsistências na BC, sendo uma das formas para detecção de incertezas abordas neste trabalho.

6.3.1 Regras de Associação Temporais Específicas

Esta subseção descreve o processo para obter as regras de associação temporais específicas (STARs - Specific Temporal Association Rules). Basicamente, após a geração das regras de

associação, o algoritmo obtém todas as STARs utilizadas para compor uma determinada regra de associação. Cada uma possui a informação da *data inicial* e *final* em que a STAR ocorreu.

Antes de descrever o processo de obtenção das STARs, alguns conceitos e definições serão introduzidos. Como já mencionado, grandes BCs como a do NELL são compostas por fatos de uma categoria específica e pelas relações entre esses fatos. Seja (c_1, c_2,..., c_n) um conjunto de categorias *C*. Cada categoria *c* tem um subconjunto de *I* (conjunto de itens de uma base de dados D). Assim, (ic_1, ic_2,..., ic_n) são os conjuntos de itens para cada categoria (c_1, c_2,..., c_n), respectivamente. *Atleta, Esporte, LigaEsportiva* são exemplos de categorias ou entidades. *Futebol Americano, basquete* e *baseball* pertencem à categoria *Esporte*, e *nfl, nba* e *mlb* são exemplos da categoria *LigaEsportiva*.

Definição 7 *Um item específico i em uma regra de associação é uma relação binária R(ic_i, ic_j), em que R é uma relação, (i,j) = (1,2,...,n), e $i \neq j$.*

Pela tabela 6.8, *atletaGanhaTrofeu(larry_foote, super_bowl)* e *atletaJogaEsporte(ben_ roethlisberger, Futebol Americano)* são exemplos de itens específicos.

Definição 8 *Uma regra de associação temporal específica (Specific Temporal Association Rule - STAR), é uma regra de associação formada por itens específicos contendo a informação do intervalo de tempo ou do período em que ocorreu.*

Para representar a informação temporal, este trabalho define dois pontos ou intervalos temporais: data inicial e data final, que identificam o período de tempo inicial e final, respectivamente. Neste trabalho, os anos inicial e final representam a medida de granularidade de cada STAR.

Definição 9 *Uma STAR é pontual se seu evento data_inicial é igual ao evento data_final. Uma STAR é não pontual se seu evento data_inicial é menor que sua data_final.*

A tabela 6.8 traz alguns exemplos de STARs. O Algoritmo 7 mostra como esse procedimento é efetuado. Para cada regra extraída, o algoritmo obtém todas as STARs utilizadas para gerar a regra (representado pelo método getValoresInstancia no Algoritmo 7). Cada STAR é adicionada em uma lista contendo a própria STAR, suas categorias e sua pontualidade (se é *pontual ou não*). Considere a regra *atletaGanhaTrofeu(X, super_bowl)* $\rightarrow$ *atletaJogaEsporte(X, Futebol_Americano)*, representada pela linha 1 na tabela 6.8. As linhas de 2 a 5, representam todas as STARs (do exemplo da tabela 6.1) que resultaram na geração da regra de associação.

Tabela 6.8: Exemplo de uma regra de associação e de suas regras de associação temporais específicas.

Número	Regra	Data Inicial	Data Final
1	*atletaGanhaTrofeu(X, super_bowl) → atletaJogaEsporte(X,Futebol_Americano)*		
2	*atletaGanhaTrofeu(ben_roethlisberger, super_bowl) → atletaJogaEsporte(ben_roethlisberger,Futebol_Americano)*	2004	2014
3	*atletaGanhaTrofeu(larry_foote, super_bowl) → atletaJogaEsporte(larry_foote,Futebol_Americano)*	2008	2008
4	*atletaGanhaTrofeu(drew_bledsoe, super_bowl) → atletaJogaEsporte(drew_bledsoe,Futebol_Americano)*	2001	2001
5	*atletaGanhaTrofeu(tom_brady, super_bowl) → atletaJogaEsporte(tom_brady,Futebol_Americano)*	2001	2001

Algoritmo 7: Obtem STARs

```
1 para i = 0 a numeroRegrasGeradas − 1 faça
2 |   RegraAtual = getRegraAssociacao(i);
3 |   para j = 0 a totalInstancias − 1 faça
4 |   |   InstanciaAtual =
  |   |     getValoresInstancia(j).getValoresParaCategorias(RegraAtual);
5 |   |   se possuiTodosValores(regraAtual,instanciaAtual) então
6 |   |   |   starLista.add(instanciaAtual);
7 |   |   fim
8 |   fim
9 fim
```

Isso significa que, em cada uma das instâncias dos atletas das linhas de 2 a 5, também apareceram os valores de *super_bowl* e *Futebol_Americano*. Note-se que cada STAR possui uma data inicial e final. STARs que possuem data inicial igual a final são consideradas *pontuais* (regras 3, 4 e 5). Caso a data inicial seja menor que a data final, a regra temporal específica é considerada *não pontual* (regra 2).

Esse procedimento é feito para todas as regras geradas. A pontualidade de cada STAR é importante para ser utilizada pelas métricas probabilísticas, com o intuito de descobrir incertezas na BC.

6.3.2 Métricas Probabilísticas

A pontualidade de cada STAR é utilizada pelas métricas probabilísticas no processo de busca por possíveis inconsistências na BC. Duas métricas foram desenvolvidas:

1. *Métrica Probabilística 1* (MP1): A MP1 consiste em identificar a porcentagem de STARs que são pontuais e/ou não pontuais para uma regra de associação. Desse modo, ela irá apontar se uma regra tem característica pontual ou não pontual, sugerindo verificar as

Tabela 6.9: Regras de Associação para popular a BC do NELL

Número	Regra de Associação
1	*atletaGanhaTrofeu(X, super_bowl) → atletaJogaEsporte(X, Futebol Americano)*
2	*atletaJogaLiga(X, nba) → atletaJogaEsporte(X, Basketball)*
3	*atletaJogaTime(X, yankees) → atletaJogaLiga(X, mlb)*
4	*atletaJogaLiga(X, nhl) → atletaJogaEsporte(X, hockey)*
5	*atletaGanhaTrofeu(X, australian_open) → atletaJogaEsporte(X, tenis)*

STARs que não o são;

2. *Métrica Probabilística 2* (MP2): A MP2 consiste em identificar a porcentagem de STARs que são pontuais e/ou não pontuais com base nos *domínios (categorias)* envolvidos em uma regra de associação. A MP2 irá auxiliar na decisão da MP1 em caso de dúvida.

Para calcular se uma regra de associação é pontual ou não, realiza-se o cálculo da porcentagem de STARs pontuais e não pontuais. A equação 6.2 traz o cálculo da porcentagem de uma STAR para descobrir se ela é pontual e a equação 6.3 para descobrir se é não pontual. Nas equações, p representa o total de STARs pontuais, np o total de não pontuais e ts o total de STARs.

$$pontual = (p/ts) * 100 \tag{6.2}$$

$$naopontual = (np/ts) * 100 \tag{6.3}$$

Um evento pontual só acontece se a *data_inicial* e *data_final* são as mesmas. Se MP1 e MP2 identificam regras de associação como sendo pontuais, todas as STARs que não são pontuais são detectadas como possíveis inconsistências. Considere agora as tabelas 6.9 e 6.10, para a exemplificação do funcionamento das métricas probabilísticas. A tabela 6.9 traz regras de associação que são utilizadas para popular a BC do NELL, e a tabela 6.10 traz as STARs que resultaram nas regras da tabela 6.9. Considere agora a regra 1 da tabela 6.9 e suas STARs (1, 2 e 3) da tabela 6.10. As STARs 2 e 3 são *pontuais* e a STAR 1 é *não pontual*. Se essas foram as únicas STARs utilizadas para gerar a regra 1, a MP1 irá sugerir para investigar a STAR 3, uma vez que a regra de associação 1 possui característica *pontual*.

No entanto, caso uma regra de associação possua uma característica *não pontual*, uma STAR *pontual* não significa uma possível inconsistência, já que eventos *não pontuais* podem iniciar e terminar em um mesmo ano. Considerando a regra de associação 3 da tabela 6.9, tem-se que, pela tabela 6.10, apenas a STAR 6 (*não pontual*) é exemplo de STAR utilizada para gerar a regra de associação. Contudo, um atleta pode jogar por uma equipe em um único ano

Tabela 6.10: STARs utilizadas para formar as regras de associação da tabela 6.9

Número	STAR	Regra na tabela 6.9
1	*atletaGanhaTrofeu(ben_roethlisberger, super_bowl) → atletaJogaEsporte(X, Futebol Americano)(2004,20014)*	1
2	*atletaGanhaTrofeu(favre, super_bowl) → atletaJogaEsporte(X, Futebol Americano) (2008,2008)*	1
3	*atletaGanhaTrofeu(peyton_manning, super_bowl) → atletaJogaEsporte(X, Futebol Americano) (2007,2007)*	1
4	*atletaJogaLiga(pau_gasol, nba) → atletaJogaEsporte(X, Basketball)(2008,2014)*	2
5	*atletaJogaLiga(paul_pierce, nba) → atletaJogaEsporte(X, Basketball)(2007,2013)*	2
6	*atletaJogaTime(lou_piniella, yankees) → atletaJogaLiga(X, mlb)(1974,1984)*	3
7	*atletaJogaLiga(sidney_crosby, nhl) → atletaJogaEsporte(X, hockey)(2005,2014)*	4
8	*atletaGanhaTrofeu(kim_clijsters, australian_open) → atletaJogaEsporte(kim_clijsters,tenis) (2011, 2011)*	5
8	*atletaGanhaTrofeu(andre_agassi, australian_open) → atletaJogaEsporte(andre_agassi,tenis) (1995, 1995)*	5
10	*atletaGanhaTrofeu(andre_agassi, australian_open) → atletaJogaEsporte(andre_agassi,tenis) (2000, 2000)*	5
11	*atletaGanhaTrofeu(pete_sampras, australian_open) → atletaJogaEsporte(andre_agassi,tenis) (1994, 1994)*	5

para uma determinada liga, o que não significa que essa é uma regra de associação *pontual*.

MP1 iria sugerir a regra de associação 3 como *pontual* já que possui mais STARs *pontuais* do que *não pontuais*. Mas, como explicado, isso não significa uma inconsistência.

Logo, dependendo dos dados armazenados, essa situação pode ocorrer recorrentemente. Para solucionar esse caso e identificar corretamente a pontualidade de uma regra de associação, no primeiro ciclo de iterações, um estudo é realizado envolvendo as categorias do subconjunto utilizado para verificar quais são *pontuais* ou *não pontuais*. Isso é efetuado para evitar a interpretação incorreta pelas métricas MP1 e MP2. Nos próximos ciclos de iterações, as métricas trabalham com base nas características de pontualidade das iterações anteriores, minimizando esse problema. Para o exemplo das tabelas 6.9 e 6.10, a categoria *Troféu* é pontual e as demais não pontuais.

Como dito anteriormente, a métrica MP2 é usada para confirmar se a pontualidade de uma regra de associação identificada pela MP1 está correta. Para isso, ela considera todas as regras de associação que possuem as mesmas categorias no lado antecedente e consequente da regra. Por exemplo, as regras de associação 2 e 4 da tabela 6.9 possuem os mesmos domínios (Liga no lado antecedente, e esporte no consequente). O mesmo ocorre para as regras 1 e 5, que possuem troféu e esporte como categorias nos antecedentes e descendentes das regras, respectivamente. Pela tabela 6.10, as STARs 4 e 5 são utilizadas para gerar a regra de associação 2, e a STAR 7 para gerar a regra 4. Nesse exemplo, todas as STARs são *não pontuais*, e a MP2 irá confirmar se uma regra de associação envolvendo esses domínios é *não pontual*.

Considerem-se agora as seguintes STARs:

STAR1:
atletaGanhaTrofeu(LeBron_James, campeonato_nba) → atletaJogaEsporte(LeBron_James, Bas-

ketball) (2010,2014)
STAR2:
atletaGanhaTrofeu(Stephen_Curry, campeonato_nba) $\rightarrow$ *atletaJogaEsporte(Stephen_Curry, Basketball) (2015,2015).*

Nesse caso, tem-se duas STARs, uma *pontual* e outra *não pontual*. Porém, as regras de associação 1 e 5 da tabela 6.9 são consideradas *pontuais* e envolvem os mesmos domínios, embora sejam regras diferentes. A MP2 procura por regras de associação com os mesmos antecedentes/descendentes e verifica suas pontualidades. Para o exemplo em questão, a MP2 irá sugerir para investigar a STAR1 pois regras de associação como *atletaGanhaTrofeu(X, Y)* $\rightarrow$ *atletaJogaEsporte(X, Z)* tendem a ser *pontuais*. Como a STAR1 possui os mesmos domínios, ela também tende a ser pontual, e o algoritmo sugere investigar a STAR1.

Ao final de cada iteração, obtém-se um conjunto de STARs que serão investigadas para descobrir possíveis inconsistências

6.3.3 Correção de Inconsistências

AS STARs sugeridas pelas métricas probabilísticas MP1 e MP2 para serem investigadas são analisadas e corrigidas quando necessário. Este trabalho apresenta duas formas possíveis para a análise e correção:

1. Realizar a análise e correção das inconsistências manualmente, de modo supervisionado;
2. Utilizar o CL para analisar e corrigir as inconsistências.

O componente CL, como previamente mencionado, faz uso do Yahoo answers para validar cada STAR (assim como para validar as regras de associação utilizadas para popular a BC do NELL). Se ele não for capaz de analisar e corrigir, é utilizado esforço manual.

Em ambos os casos, as possíveis inconsistências sugeridas pelas métricas probabilísticas são investigadas. Considerando a STAR1 exemplificada anteriormente, é possível que o atleta *LeBron James* tenha vencido o campeonato da *nba* em um ou mais anos entre a data inicial e final que a STAR1 possui, mas não em todos os casos. Isso é uma possível inconsistência.

Neste caso, tem-se uma STAR não pontual para uma regra de associação pontual. Após a análise, verificou-se que o atleta ganhou 2 vezes o campeonato da nba naquele intervalo temporal. A tabela 6.11 mostra como o processo de correção é realizado. A primeira linha representa a

Tabela 6.11: Correção das instâncias para o atleta Lebron James

Número	Atleta	Estádio	Time	Liga	Esporte	Troféu	Data inicial	Data final
1	LeBron James	quicken loans arena	mv	nba	Basquete	campeonato nba	2010	2014
2	LeBron James	quicken loans arena	mv	nba	Basquete	campeonato nba	2012	2012
3	LeBron James	quicken loans arena	mv	nba	Basquete	campeonato nba	2013	2013

instância original do subconjunto da BC. As linhas 2 e 3 correspondem às instâncias que foram corrigidas. Note que, nas linhas 2 e 3, os valores para a data *inicial* e *final* são *pontuais*, o que mostra que a inconsistência se devia ao intervalo de tempo incorreto para a categoria Troféu.

Todavia, se a correção for realizada no mesmo conjunto de dados, conforme o ilustrado na tabela 6.11, dois problemas podem ocorrer:

1. O conjunto de dados pode tender a possuir característica *pontual*, o que irá resultar em mais regras de associação *pontuais*;
2. O valor do suporte dos itemsets pode passar a ser maior.

Para solucionar o primeiro problema, foi criada uma tabela adicional, a qual é denominada *tabela pontual*. Ela irá conter os dados de todas as instâncias corrigidas que eram *não pontuais* mas deveriam ser como nas linhas 2 e 3 da tabela 6.11. A solução elaborada é manter a linha original no conjunto principal de dados, com o intervalo original como *não pontual*, e armazenar as instâncias corrigidas na *tabela pontual*. Para o exemplo, as linhas 2 e 3 estariam na *tabela pontual*.

Observe que, se somente as linhas 2 e 3 fossem adicionadas ao conjunto de dados original, ele teria mais instâncias *pontuais*, o que iria tendenciar tanto o conjunto de dados quanto as regras de associação a serem *pontuais*. Dessa forma, a *tabela pontual* é uma solução eficiente para esse problema, mantendo a instância original (que possui outras categorias que são não pontuais como o time que o atleta joga) e inserindo as correções na tabela pontual.

A partir da segunda iteração do primeiro ciclo, o algoritmo irá utilizar essa tabela se a regra de associação tiver característica *pontual*, que irá adicionar as STARs (relacionadas às instâncias corrigidas) à regra de associação que está sendo verificada.

Além disso, ao se criar a *tabela pontual* auxiliar para armazenar as instâncias que continham erros e eram classificadas como *não pontuais*, os valores do suporte dos itemsets não é afetado, o que auxilia a resolver o segundo problema. Se as linhas 2 e 3 da tabela 6.11 fossem adicionadas

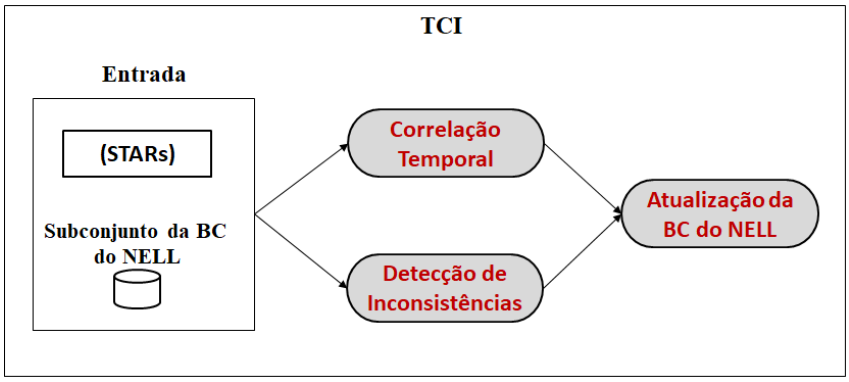

Figura 6.4: Arquitetura do componente TCI

ao conjunto de dados original, alguns itemsets teriam seus valores de suporte maiores, como *quicken loans arena, nba* e *basquete*.

É possível observar que as linhas de 1 a 3 possuem os mesmos valores, exceto por suas datas *iniciais* e *finais*. Se um de seus valores fosse diferente, as instâncias não seriam consideradas iguais e deveriam estar presentes no conjunto original de dados. Um exemplo é quando um atleta vence o campeonato da nba em diferentes anos por equipes diferentes.

6.4 TCI - Temporal Correlation Inference

O componente TCI é dividido em dois subcomponentes: (i) *Correlação Temporal (Temporal Correlations)*, que procura por correlações temporais entre as instâncias com o intuito de popular a BC do NELL, e (ii) *Detecção de Inconsistência (Inconsistency Detection)* que possui o propósito de detectar inconsistências entre as instâncias da BC. A figura 6.4 ilustra a arquitetura do componente TCI. Basicamente, o TCI recebe como entrada as STARs geradas pelo componente TARE e o subconjunto de dados do NELL. Ambos os subcomponentes são executados em paralelo. Caso seja encontrada uma correlação para popular a BC, ela é atualizada. Por outro lado, se uma possível inconsistência é encontrada, ela é analisada e corrigida (atualizando a BC do NELL) quando necessário.

Para obter a correlação temporal entre as instâncias, o TCI usa um método baseado no intervalo temporal de Allen. No entanto, algumas mudanças foram realizadas para comparar dois intervalos temporais. Assim, o TCI possui três contribuições deste trabalho:

- Modificação do intervalo temporal de Allen (TIA) para efetuar as técnicas de correlação temporal e a detecção de inconsistências;
- Identificação de inconsistências com base nas correlações temporais entre as STARs ex-

traídas e outras instâncias na base de conhecimento;

- Identificação de correções temporais entre as STARs extraídas e outras instâncias para popular a base de conhecimento.

6.4.1 Adpatação do Intervalo Temporal Algébrico de Allen

Tanto o subcomponente de *Correlação Temporal* quanto o de *Detecção de Inconsistência* usam o intervalo temporal algébrico modificado de Allen. No processo de encontrar correlações temporais ou de detectar inconsistências, o procedimento compara dois pontos temporais (no caso, data inicial e final). Considere *A (a-, a+)* e *B (b-, b+)* como dois pontos ou intervalos temporais. Neste trabalho, *a- e b-* são as *datas iniciais*, e *a+ e b+* as *finais* de *A* e *B*, respectivamente. Logo, o algoritmo verifica se a *data inicial a-* começa antes, ao mesmo tempo ou após *b-*. O mesmo processo é feito para as *datas finais a+ e b+*.

A tabela 6.12 mostra todas as nominações usadas neste trabalho (sem as possibilidades reversas). Dois novos intervalos temporais foram criados:

1. *startLessEndLess*, que unifica as relações temporais *meets* e *overlaps* de Allen;
2. *startMoreEndMore*, que unifica as relações temporais reversas *meets_i* e *overlaps_i* de Allen.

Essa é a razão pela qual as relações temporais de Allen são reduzidas de 13 para 11 se comparadas às propostas neste trabalho. Pelas relações *meets* e *overlaps* de Allen, $a- < b-$ e $a+ < b+$, o que indica que a data inicial do período A é menor que a de B, e a data final de A também é menor do que a de B (unificadas na relação *startLessEndLess* neste trabalho). No modo inverso, o reverso de *meets* e *overlaps* são unificados na relação *startMoreEndMore*.

No processo de encontrar correlações temporais ou de detectar inconsistências, as relações temporais *after* e *before* são as únicas que não possuem uma intersecção temporal entre dois intervalos de tempo. Em todos os outros casos, é possível atualizar a BC, populando-a com novas instâncias com base nas correlações temporais, ou detectar possíveis inconsistências.

6.4.2 Correlação Temporal

Este subcomponente trabalha em paralelo com o de detecção de inconsistência. O principal propósito é encontrar instâncias que possuam alguns valores em comum durante uma das hipóteses modificadas de Allen e atualizar a BC com os valores não presentes na outra.

Tabela 6.12: Intervalo Temporal de Allen modificado

Relações Temporais	Condições em ambos pontos do intervalo	Representação
before(A,B)	a- < b-, a- < b+	AAAA
	a+ < b-, a+ < b+	BBBBBBB
after(A,B)	a- > b-, a- > b+	AAAA
	a+ > b-, a+ > b+	BBBB
starts(A,B)	a- = b-, a- < b+	AAAA
	a+ > b-, a+ < b+	BBBBBBB
finishes(A,B)	a- > b-, a- < b+	AAAA
	a+ > b-, a+ = b+	BBBBBBB
during(A,B)	a- > b-, a- < b+	AAAA
	a+ > b-, a+ < b+	BBBBBBB
equal(A,B)	a- = b-, a- < b+	AAAAA
	a+ > b-, a+ = b+	BBBBB
startLessEndLess(A,B)	a- < b-, a- < b+	AAAAA
	a+ > b-, a+ < b+	BBBBBB
startMoreEndMore(A,B)	a- > b-, a- < b+	AAAAAAA
	a+ > b-, a+ > b+	BBBBB

Tabela 6.13: Exemplo de instâncias

Número	**Atleta**	**Técnico**	**Esporte**	**Liga Esportiva**	**Troféu**	**Time**	**data inicial**	**data final**	Intervalo Temporal
1	Pau Gasol	mv	basquete	nba	campeonato nba	la lakers	2008	2014	
2	Kobe Bryant	Phil Jackson	mv	mv	mv	la lakers	1996	2015	during
3	Lamar Odom	mv	mv	nba	campeonato nba	mv	2004	2011	startMoreEndMore
4	Manu Ginobili	mv	basquete	nba	campeonato nba	spurs	2002	2014	finishes
5	Tim Duncan	Gregg Popovich	mv	nba	campeonato nba	spurs	1997	2015	during

Considere-se que o TARE tenha gerado a seguinte *STAR*:

STAR1:
atletaJogaTime(Pau_Gasol, la_lakers) $\rightarrow$ *atletaJogaLiga(Pau_Gasol, nba) (2008,2014).*

Primeiro, o componente TCI procura por valores em outras categorias relacionados com o*atleta Pau Gasol*, contendo as mesmas *data_inicial* e *data_final*. Isso é feito para obter as instâncias desse atleta relacionadas com esse período temporal. A tabela 6.13 mostra alguns dos seus valores e instâncias relacionados que foram retornados pelo algoritmo. A última coluna da tabela representa os intervalos temporais usados entre a instância da linha 1 e as demais. O único valor que o atleta Pau Gasol não possui corresponde ao valor de seu técnico (valores ausentes são representados por um *mv* na tabela). O algoritmo 8 possui o pseudo-código do subcomponente Correlação Temporal.

Para cada STAR gerada e armazenada em uma lista (listaDeSTARs no algoritmo 8), o subcomponente Correlação Temporal seleciona todas as instâncias na BC que estão relacionadas com a STAR corrente (método getInstancesFromStart()) e que possuem algum valor em co-

Algoritmo 8: Algoritmo de Correlação Temporal

Entrada: *listaDeSTARs*
Saída:

```
1 para i = 0 a listaDeSTARs − 1 faça
2 |   starAtual = getStarAtual(i);
3 |   arrayInst = getInstancesFromStart(starAtual);
4 |   para j = 0 a arrayInst − 1 faça
5 |   |   se hasCategoryMissingValue(arrayInst(j)) então
6 |   |   |   update(arrayInst(j), starAtual);
7 |   |   fim
8 |   fim
9 fim
```

mum. Depois, para cada instância retornada, ele compara se uma determinada categoria possui um valor ausente (método hasCategoryMissingValues()), e a atualiza com os valores da STAR atual. A tabela 6.13 traz as instâncias relacionadas à *STAR1* que possuem *Troféu* = *campeonato nba* e *Time* = *la lakers* como valores comuns. Para esse exemplo, o algoritmo (i) obtém os valores das categorias de Pau Gasol, (ii) verifica onde existe um valor ausente para as categorias correspondentes nas outras instâncias, e (iii) atualiza as instâncias com os valores de Pau Gasol sempre que um valor ausente ocorre.

Um ponto importante a ressaltar é que nem todas as categorias são relevantes para a consulta executada utilizando a linha 1 da tabela 6.13. Por exemplo, se uma pessoa joga pela liga *nba* durante um intervalo temporal que está intersectado em algum momento com o de Pau Gasol, não significa que a pessoa joga no mesmo time ou possui o mesmo técnico que Gasol. Portanto, é necessário que o subcomponente verifique quais categorias são relevantes ou não para serem utilizadas na consulta. Para isso, o Correlação Temporal verifica em uma tabela quais categorias são irrelevantes para a consulta. Semelhantemente ao que ocorre no componente TARE (que faz um estudo de quais domínios possuem característica pontual para o primeiro ciclo de iterações), uma análise é realizada para identificar quais são as categorias que são irrelevantes e não devem ser utilizadas na consulta.

Considerando a tabela 6.13, uma atualização pode ser feita para a linha 2. Note que ela é retornada porque ambos os atletas (linhas 1 e 2) possuem o mesmo time esportivo. O atleta *Kobe_Bryant* possui valores ausentes relacionados às categorias *Esporte, Liga Esportiva e Troféu*. Todas são atualizadas com os valores do jogador *Pau_Gasol* (basquete, nba e campeonato nba). Isso contribui para alimentar a base de conhecimento do NELL e diminuir a quantidade de valores ausentes. Ao comparar com o método modificado de Allen, as linhas 1 e 2 possuem a relação temporal *during*, uma vez que o período de tempo de 1 está incluso em 2.

Além de procurar por instâncias com alguns valores em comum, ele também verifica se um domínio específico possui uma característica *pontual*. Esse procedimento é feito com a verificação da porcentagem de STARs *pontuais* que determinado domínio possui. Se um domínio é identificado como sendo *pontual*, uma atualização só pode ser efetuada se os eventos de data_inicial e data_final são idênticos para ambas as instâncias, ou se possuírem mais de um valor em comum. Caso contrário, uma possível inconsistência é sugerida pelo subcomponente de Detecção de Inconsistência (linha 3). Esse e outros casos de inconsistências são descritos a seguir. Pela tabela 6.13, as linhas de 3 a 5 são casos em que o subcomponente de detecção de inconsistência atua.

6.4.3 Detecção de Inconsistência

A maioria das pesquisas que trabalham com ruídos em suas bases de conhecimento não levam em consideração a informação temporal. Este trabalho apresenta uma inovação no processo de detecção de imprecisões na BC, que se baseia na identificação de diferenças entre instâncias que possuem alguns valores em comum (mas não todos) durante um período de tempo específico, com o intuito de encontrar dados com erros na BC. Além desse processo, como foi citado da subseção anterior, também é possível que um valor comum entre duas instâncias tenha uma imprecisão temporal. Isso ocorre se o valor em questão pertence a um domínio *pontual* (método descrito anteriormente) e pelo menos um dos eventos envolvendo as duas instâncias seja *não pontual*. Logo, existem duas possibilidades de se encontrarem inconsistências.

O método de correlação temporal atualiza a BC ao encontrar valores comuns entre duas ou mais instâncias específicas. Neste momento, o subcomponente de detecção de inconsistências está interessado em encontrar ruídos nos dados já armazenados na BC. Ambas as técnicas são semelhantes. Mas, em vez de procurar por possíveis valores comuns entre as instâncias, ele busca por categorias que possuem um valor em comum entre os requisitados pela consulta e, especialmente, pela diferença entre eles.

O algoritmo 9 traz o pseudo-código do procedimento para detecção de inconsistências, o qual é bem semelhante ao de correlação temporal. Contudo, em vez de procurar por um valor em comum, o algoritmo procura por diferenças em instâncias que possuem algum valor em comum para os períodos temporais em questão (método hasDifferentValues no algoritmo 9). Caso seja detectada uma possível inconsistência, ela é armazenada e será investigada para futura correção.

Assim como a Correlação Temporal, a Detecção de Inconsistências não considera algumas categorias durante o processo de identificação de erros. Por exemplo, imagine que exista a categoria *lesão* (a qual informa se um atleta se machucou). Se diversos atletas tiverem valores

Algoritmo 9: Algoritmo de Detecção de Inconsistências

```
   Entrada: listaDeSTARs
   Saída:
1  para i = 0 a listaDeSTARs − 1 faça
2  |  starAtual = getStarAtual(i);
3  |  arrayInst = getInstancesFromStart(starAtual);
4  |  para j = 0 a arrayInst − 1 faça
5  |  |  se hasDiferentValues(arrayInst(j)) então
6  |  |  |  inserePossivelIncons(arrayInst(j), starAtual);
7  |  |  fim
8  |  fim
9  fim
```

diferentes para a categoria *lesão*, não significa que há uma inconsistência na base de conhecimento, uma vez que dois ou mais *atletas* podem atuar por um mesmo time no mesmo período, mas com um tipo diferente de lesão ou nenhuma (valor ausente).

Pela tabela 6.13, as linhas 4 e 5 apresentam inconsistências se comparadas com a 1 (intervalos temporais *finishes* e *during*, respectivamente). Isso significa que em uma delas (ou em ambas) existe uma possível imprecisão. Logo, o algoritmo sugere investigar ambas para que se verifique se há e onde está a imprecisão (que pode ser por um fato aprendido erroneamente ou por período temporal incorreto). Para os dois casos do exemplo, as instâncias das linhas 3 e 4 foram retornadas pela consulta por causa do valor comum da categoria *Troféu* = *campeonato_nba*, mas o domínio *Time* possui valores diferentes, o que indica um conflito entre essas duas instâncias. Pode-se dizer que os atletas das linhas 4 e 5 ganharam a liga *nba* em um ou mais anos entre as suas datas iniciais e finais, mas não em todos os anos. Contudo, o presente trabalho procura obter informações precisas para evitar a propagação de erros. Nos exemplos da tabela 6.13, não é possível saber o ano exato em que aqueles atletas ganharam o campeonato da *nba*. Se este procedimento não fosse efetuado, o algoritmo atualizaria a base de conhecimento com informação errada. Por exemplo, o algoritmo poderia atualizar o time dos atletas *Manu Ginobili e Tim Duncan* para *la_lakers*, o que iria propagar informação errada para a BC. Dessa forma, este método é fundamental no processo de detecção de incertezas presentes na BC.

A linha 3 representa a outra possibilidade de busca por incertezas na BC pelo subcomponente de detecção de inconsistências. Como já mencionado, este método identifica se uma categoria contendo um valor comum entre duas instâncias é *pontual*. Caso seja, o algoritmo irá verificar as datas iniciais e finais de cada instância. Se pelo menos uma delas possuir o período temporal como *não pontual*, o algoritmo indica que há uma possível inconsistência em uma ou em ambas as instâncias. No exemplo da linha 3, o valor em comum é *campeonato_nba*

que pertence ao domínio *Troféu* e não há valores diferentes para outras categorias. No entanto, o componente TCI reconhece a categoria *Troféu* como sendo *pontual* e, então, ele verifica o período de tempo das instâncias. Para a tabela 6.13, tanto a linha 1 quanto a 3 são *não pontuais*, fazendo com que o algoritmo sugira a verificação das duas instâncias.

Desse modo, o componente TCI contribui com os dois métodos, tendo o intuito de verificar as incertezas presentes na BC. Cada possível inconsistência detectada pelo TCI é avaliada para verificar se realmente existe algum erro. Caso possua, este trabalho de doutorado propões que se faça a correção de uma das duas forma a seguir:

1. Utilizar o CL e as respostas dos usuários do Twitter para obter a informação precisa;
2. Realizar correção manual caso o CL não consiga.

Capítulo 7

EXPERIMENTOS E RESULTADOS

Esta seção traz os experimentos realizados neste trabalho de doutorado, bem como os trabalhos comparativos. A subseção 7.1 mostra os experimentos realizados envolvendo apenas regras de associação em comparação com um algoritmo tradicional de regras de associação (FP-Growth). Também é feita uma comparação com técnicas que visam à redução do número de itemsets e de regras de associação para verificar o comportamento das técnicas de eliminação de regras redundantes e irrelevantes do componente ER. A seção 7.2 ilustra os experimentos realizados envolvendo os componentes TARE e TCI. Nestes, o objetivo é verificar o comportamento de ambos os componentes na (i) detecção de inconsistências por meio das métricas probabilísticas (TARE) e do subcomponente de detecção de inconsistências (TCI), e na (ii) expansão da base de conhecimento do NELL com a correlação temporal do componente TCI. Por fim, a seção 7.3 traz a análise de desempenho do algoritmo de regras de associação, dos métodos do componente ER, e dos componentes TARE e TCI.

7.1 Experimentos Regras de Associação

Os experimentos realizados envolvendo regras de associação possuem os seguintes objetivos:

- Verificar como as regras de associação podem auxiliar a expandir uma GBC;
- Verificar como o parâmetro MSC auxilia na produção de mais regras de associação em grandes bases contendo valores ausentes;
- Analisar o impacto dos métodos de eliminação de regras redundantes e irrelevantes após a geração das regras (ER);

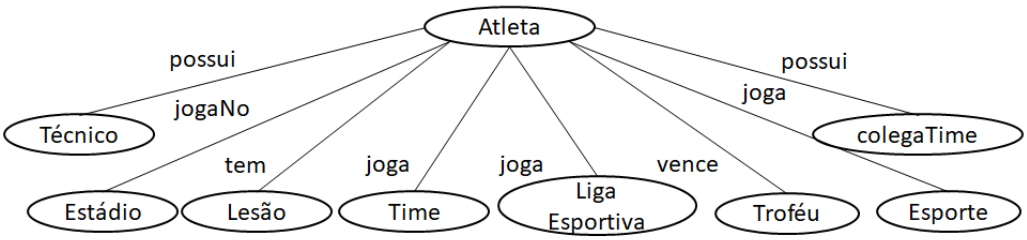

Figura 7.1: Categorias do subconjunto da BC do NELL utilizadas nos experimentos

- Mostrar a necessidade de avaliação das regras de associação geradas com grandes bases de conhecimento crescentes.

Pretende-se verificar a eficácia dos métodos propostos para geração de regras de associação em grandes bases de conhecimento crescente. Além de verificar o quanto as regras de associação auxiliam a popular GBCs, os experimentos analisam a quantidade de regras geradas devido ao parâmetro MSC, o qual foi desenvolvido para trabalhar com valores ausentes. Como já mencionado, algoritmos de regras de associação produzem muitas regras, e analisar cada uma pode ser uma tarefa árdua. Assim, os experimentos também visam a analisar o quanto os métodos para eliminação de regras diminuem o esforço na análise das mesmas. Por fim, devido à característica da BC, realizar a análise de cada regra gerada é essencial. Desse modo, após a geração do conjunto final de regras, cada uma é analisada com auxílio de outro componente do NELL, o Conversing Learning.

Para realizar os experimentos, foram utilizados suportes mínimos variando entre 0.04 e 0.01, decrescendo em 0.01 a cada execução. A confiança mínima desejada foi fixada em 0.3. Os valores de suporte são relativamente baixos devido à característica da base de conhecimento, que é composta por inúmeros dados para cada categoria. Em adição, nenhuma regra de associação foi gerada ao se utilizar suporte mínimo em 0.05. Foi utilizado um subconjunto de dados esportivos extraídos da BC do NELL, conforme a figura 7.1. Cada elipse na figura representa uma categoria da BC. O subconjunto possui fatos relacionados a essas categorias.

A estrutura ontológica do subconjunto da BC do NELL foi criada utilizando o Protégé (KNUBLAUCH et al., 2004), o qual possui todas as categorias e todos os fatos utilizados nos experimentos.

Além de verificar os resultados das técnicas desenvolvidas neste trabalho, foram realizadas avaliações empíricas com o intuito de identificar qual a melhor forma de executar o algoritmo. Dois procedimentos foram testados:

1. Validar as regras obtidas e atualizar a BC do NELL com as regras válidas no final de

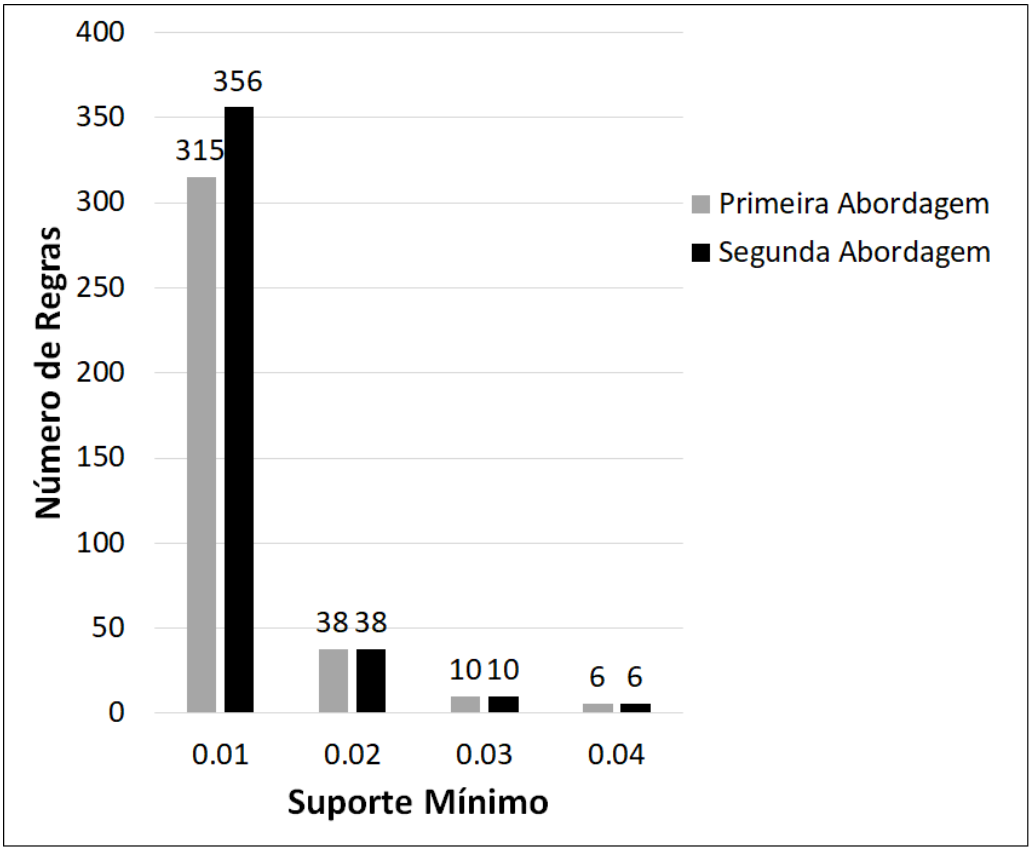

Figura 7.2: Comparação entre Primeira e Segunda Abordagem

cada ciclo completo de iteração executado pelo algoritmo em determinada base (primeira abordagem);

2. Validar as regras obtidas e atualizar a BC do NELL com as regras válidas no final de cada iteração do ciclo executado pelo algoritmo em determinada base (segunda abordagem).

Para a primeira abordagem, o algoritmo executa em uma determinada base de dados (contendo um subconjunto esportivo como o da figura 7.1) com todos os suportes mínimos (0.04, 0.03, 0.02 e 0.01), o que é considerado um ciclo completo de execução. Após a finalização de todo o ciclo, todas as regras validadas são utilizadas para atualizar a BC. Na segunda, a base é atualizada imediatamente após a execução com cada suporte mínimo. Isto é, as regras obtidas e relevantes para o suporte 0.04 são utilizadas para atualizar a BC atual. A base atualizada já é utilizada como entrada para o suporte 0.03. O mesmo é feito após a execução com suportes 0.03, 0.02 e 0.01.

Após a execução do algoritmo com as duas abordagens, o segundo procedimento obteve mais regras relevantes, principalmente com suportes mais baixos. Isso pode ser observado na figura 7.2.

Feita esta análise inicial, todos os experimentos dessa seção utilizam a segunda abordagem, ou seja, a base é atualizada no final de cada iteração do ciclo. Para melhor compreensão, os experimentos foram divididos em:

- **Fase 1**: em que são analisados o comportamento do parâmetro MSC na geração de regras e a necessidade de avaliação das regras geradas ao final de cada iteração;
- **Fase 2**: em que são realizados experimentos envolvendo as técnicas para eliminação de regras redundantes e irrelevantes.

7.1.1 Fase 1

Nesta subseção é analisada a influência do parâmetro MSC no processo de geração de regras, bem como a necessidade de análise das regras geradas. Para o primeiro caso, é feito um comparativo entre o algoritmo FP-Growth (HAN; PEI; YIN, 2000) (um algoritmo que possui o cálculo tradicional de suporte) e o proposto neste trabalho, com a introdução do parâmetro MSC. FP-Growth é baseado no *Apriori*, mas possui maior eficiência.

A figura 7.3 mostra a quantidade de regras geradas por cada algoritmo. Como pode ser observado, à medida que o valor do suporte mínimo desejado diminui a quantidade de regras de associação geradas por ambos aumenta. No entanto, devido ao fato de o parâmetro MSC gerar itemsets frequentes com suporte maior ou igual ao de algoritmos que utilizam o cálculo tradicional de suporte, mais regras são geradas ao se utilizar a abordagem proposta neste trabalho. O número de regras geradas utilizando o parâmetro MSC é maior para todos os suportes mínimos utilizados, tornando-se um número muito maior conforme o valor do suporte mínimo diminui (mais de 250 regras para o suporte de 0.01).

Com um número maior de regras geradas, uma maior quantidade de valores ausentes é preenchida, resultando em uma BC com mais dados. Dessa forma, as regras geradas devido ao parâmetro MSC auxiliam mais no processo de popular grandes bases de conhecimento do que se comparadas com as geradas por algoritmos tradicionais de extração de regras de associação.

No entanto, nem toda regra gerada é utilizada para popular a BC do NELL. Muitas das regras produzidas são irrelevantes. Assim, é necessário analisar cada regra gerada, e somente as regras válidas são utilizadas para estender a BC do NELL com dados. Para realizar a validação das regras, o componente CL (PEDRO; HRUSCHKA JUNIOR, 2012b) é utilizado. A figura 7.4 traz a quantidade de regras válidas geradas pelos dois algoritmos da figura 7.3. O comportamento é o mesmo, ou seja, à medida que o valor do suporte mínimo diminui, a quantidade de regras

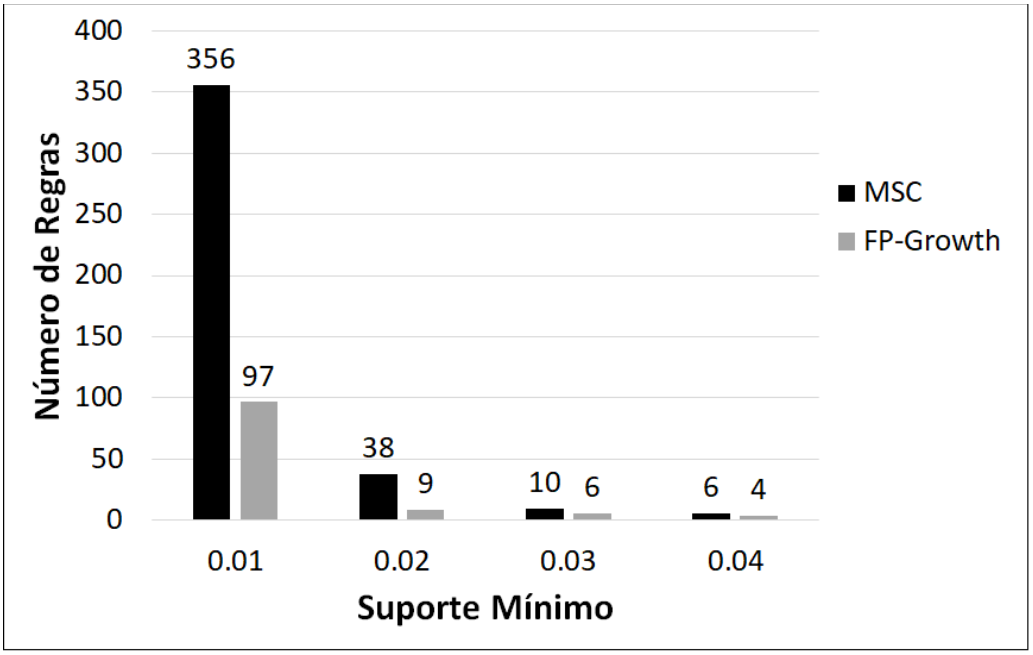

Figura 7.3: Comparação entre MSC x PF-growth

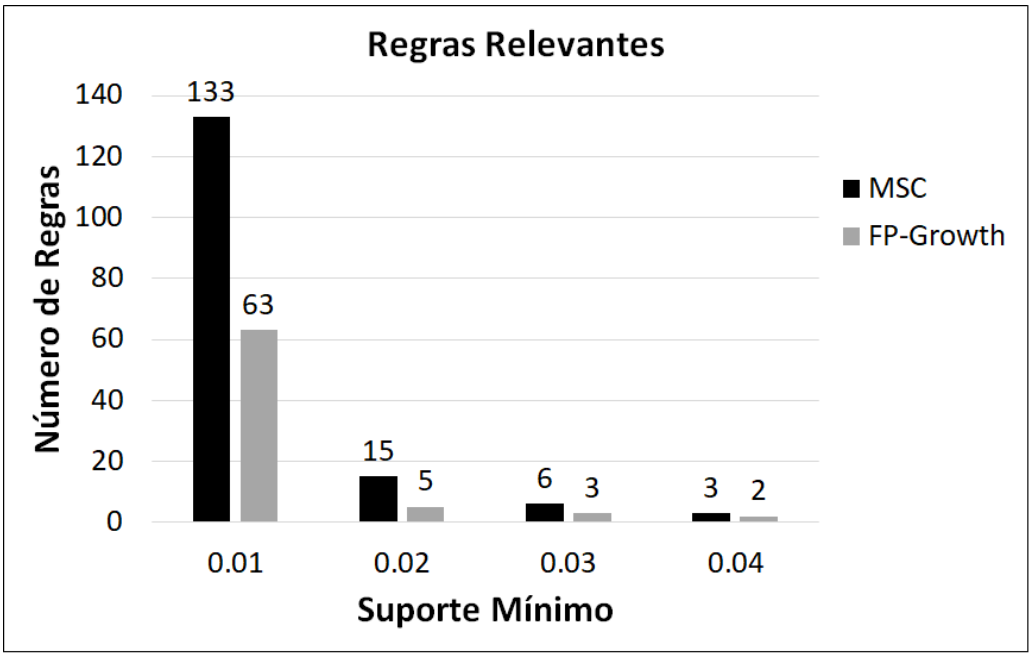

Figura 7.4: Regras Relevantes - MSC x PF-growth

aumenta.

A tabela 7.1 traz exemplos de regras relevantes e irrelevantes extraídas pelo método proposto neste trabalho para a figura 7.4. As regras 1, 2 e 4 são analisadas pelo CL e consideradas relevantes. Logo, serão utilizadas para popular a BC do NELL. Isto é, sempre que o algoritmo encontrar os valores do antecedente de uma dessas regras em uma instância, irá atualizá-la com os valores dos consequentes da regra. No entanto, segundo os usuários do CL, a regra 3 não

Tabela 7.1: Exemplo de regras extraídas

Número	Regra de Associação	Relevante
1	*atletaGanhaTrofeu(X, super_bowl)* → *atletaJogaEsporte(X, Futebol Americano)*	Sim
2	*atletaJogaLiga(X, nba)* → *atletaJogaEsporte(X, Basquete)*	Sim
3	*atletaJogaEsporte(X, Basquete)* → *atletaJogaLiga(X, nba)*	Não
4	*atletaJogaLiga(X, nfl)* → *atletaJogaEsporte(X, Futebol Americano)*	Sim

é relevante para o processo de preencher os valores ausentes. Imagine que os componentes do NELL descubram que um atleta brasileiro joga o esporte *Basquete*, mas só tenha jogado no Brasil a liga *nbb* (valor ainda não presente na BC do NELL). Caso a regra 3 fosse considerada no processo de popular a BC, o algoritmo poderia, erroneamente, preencher com o valor de *nba* para a liga que o atleta em questão joga.

7.1.2 Fase 2

A fase 2 tem como intuito amenizar o esforço ao avaliar cada regra extraída. Conforme o ilustrado pelas figuras 7.3 e 7.4, muitas regras extraídas são irrelevantes e não podem ser utilizadas para preencher a BC, uma vez que irão propagar erros na mesma. Além disso, muitas regras extraídas podem ser consideradas redundantes. Para resolver esses problemas, o presente trabalho desenvolveu o componente ER, que possui dois métodos para eliminar (i) regras redundantes e (ii) irrelevantes.

Como descrito no capítulo 6, são eliminadas super-regras antecedentes consideradas redundantes e super-regras consequentes consideradas irrelevantes, o que é diferente de alguns métodos descritos no capítulo 2, sendo essas ações contribuições deste trabalho.

Para realizar esses experimentos, são utilizados os mesmos conjuntos de dados dos experimentos realizados para analisar os componentes TARE e TCI (descritos na próxima seção). Assim, além de atualizar a BC após cada iteração do ciclo, o subconjunto utilizado também possui a correção das inconsistências encontradas, bem como as atualizações realizadas pelo componente TCI. Os valores de suportes mínimos e da confiança são os mesmos dos realizados na fase 1.

Três ciclos de iteração foram executados. Cada um com um subconjunto de dados estendido com fatos da NELL para o subconjunto de dados correspondente. Considere **subconjunto de dados 1**, **subconjunto de dados 2**, e **subconjunto de dados 3** aqueles usados nos ciclos 1, 2 e 3, respectivamente. Desse modo, três experimentos foram executados, comparando a quantidade de regras extraídas pelo algoritmo original (utilizando o parâmetro MSC) com a quantidade obtida após aplicar os métodos de eliminação de regras redundantes e irrelevantes:

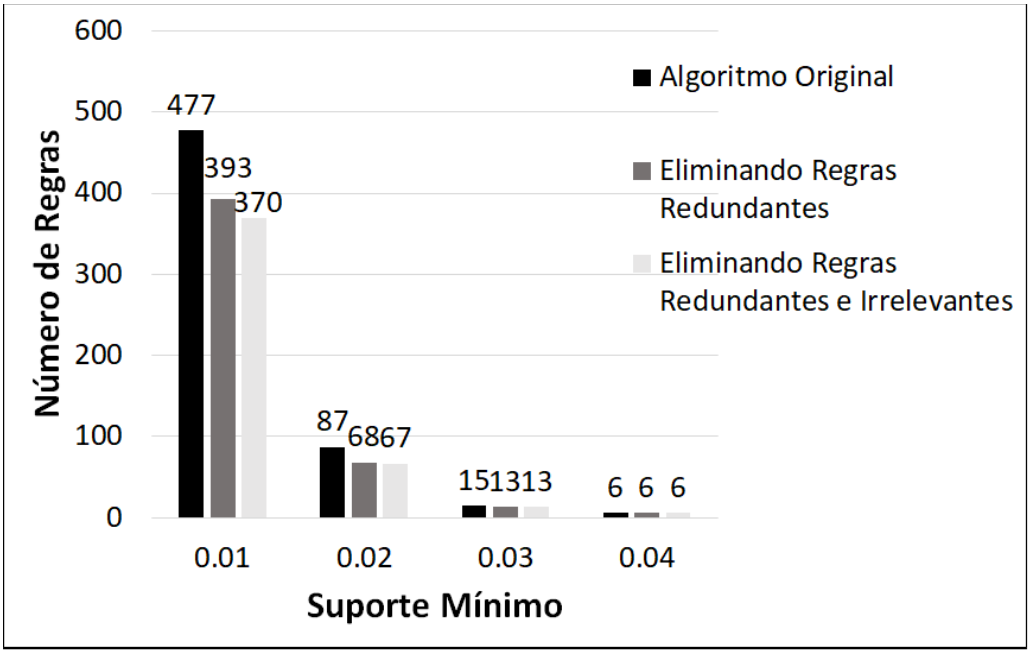

Figura 7.5: Número de Regras extraídas pelo Experimento 1

- **Experimento 1**: subconjunto de dados 1 e ciclo 1;
- **Experimento 2**: subconjunto de dados 2 e ciclo 2;
- **Experimento 3**: subconjunto de dados 3 e ciclo 3.

Também é realizada uma comparação com os algoritmos FP-Growth, CHARM e FPMax em dois aspectos:

- O número de itemsets frequentes gerados;
- A porcentagem de valores ausentes preenchidos (comparado somente com FP-Growth e CHARM).

A figura 7.5 traz uma comparação entre (i) o número de regras de associação extraídas pelo algoritmo proposto neste trabalho sem os métodos de eliminação de regras, e (ii) a quantidade de regras descobertas ao se aplicar somente o método de redundância e (iii) ambos os métodos para eliminação de regras. Com a utilização de ambos os métodos, foram obtidas cerca de 22.43% menos regras do que o algoritmo original no **Experimento 1**.

Como pode ser observado na figura 7.5, o número de regras de associação aumenta consideravelmente ao reduzir o valor do *suporte mínimo*. Com o maior valor para o *suporte mínimo* (0.04), apenas regras envolvendo 2-itemsets (itemsets de tamanho 2) foram geradas. Logo, os

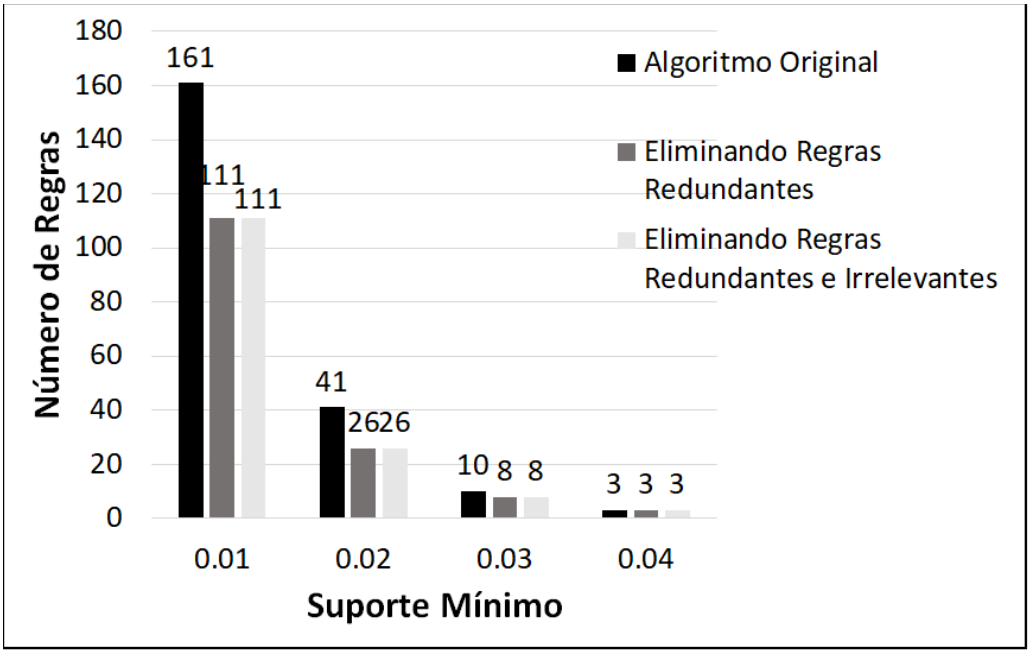

Figura 7.6: Número de Regras Relevantes extraídas pelo Experimento 1

métodos não foram aplicados para reduzir a quantidade de regras, uma vez que eles têm a função de eliminar *super ARs* redundantes e *super CRs* irrelevantes, as quais possuem pelo menos dois itens no lado antecedente e consequente, respectivamente. Mas, conforme o *suporte mínimo diminui*, os métodos para eliminação são aplicados, contribuindo para reduzir o número de regras a serem avaliadas.

Assim como feito na fase 1, todas as regras descobertas são avaliadas pelo CL. As que forem consideradas irrelevantes com o *suporte mínimo* de 0.04 são usadas em todas as outras iterações e nos próximos ciclos (**Experimentos 2** e **3**) pelo método de eliminação de regras redundantes. O mesmo é feito para as regras irrelevantes descobertas com *suporte mínimo* de 0.03 e 0.02. Contudo, as regras irrelevantes descobertas com *suporte mínimo* de 0.01 só serão utilizadas em futuros ciclos de iterações.

Pela figura 7.5, com suporte 0.03, a quantidade de regras ao eliminar somente as regras redundantes e ambas regras redundantes e irrelevantes é a mesma. Nesse caso, apenas super ARs redundantes foram excluídas. Já com os valores de *suporte mínimo* de 0.02 e 0.01, principalmente, são eliminadas super ARs redundantes e super CRs irrelevantes.

A figura 7.6 mostra apenas o número de regras relevantes extraídas no **Experimento 1**. Note-se que o algoritmo original (sem os novos métodos) trouxe mais regras relevantes em comparação com os métodos para eliminação de redundância e irrelevância. Isso é resultado da quantidade de regras redundantes a mais que o algoritmo original possui (50 para o suporte mínimo de 0.01) se comparado com as abordagens para eliminação de regras introduzidas neste

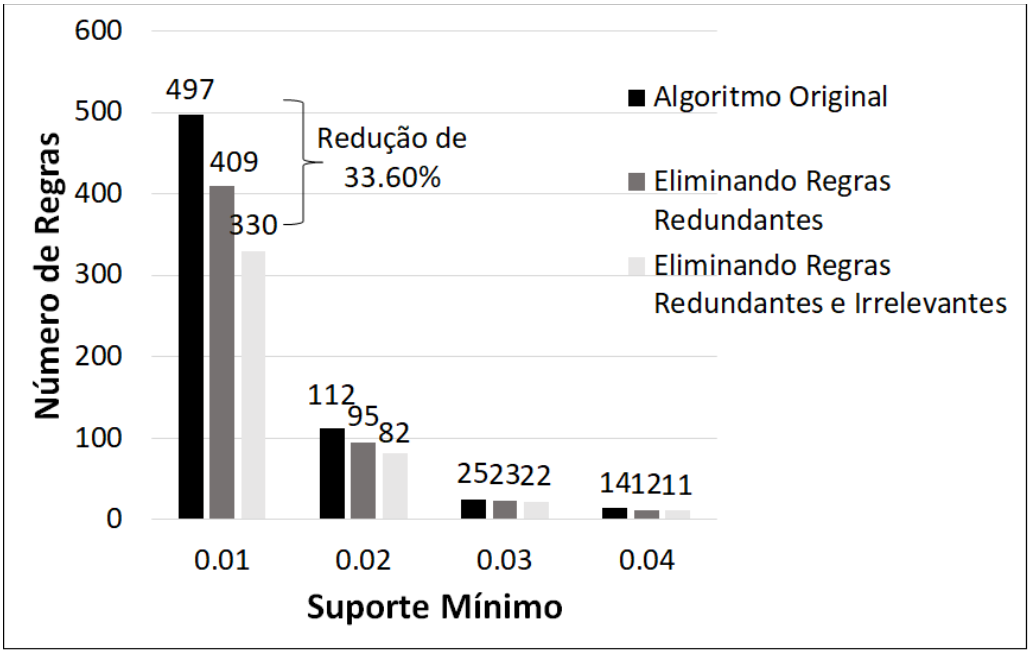

Figura 7.7: Número de Regras extraídas pelo Experimento 2

trabalho. Também é possível observar que o número de regras relevantes obtidas por ambos os métodos de redundância e irrelevância é o mesmo. Pela figura 7.5, a quantidade extraída pelo método de eliminação de regras redundantes é um pouco maior que a obtida ao se eliminar as regras irrelevantes também, já que o primeiro método ainda não havia eliminado regras de associação irrelevantes.

É importante salientar que a aplicação do componente ER, além de diminuir a quantidade de regras, popula a BC do subconjunto utilizado com a mesma quantidade de dados que o algoritmo original (sem os métodos de redução de regras), mostrando-se uma ferramenta eficiente para facilitar a análise das regras.

Os resultados do **Experimento 2** estão na figura 7.7. O conjunto de dados usado foi estendido em, aproximadamente, 10%, uma vez que o NELL possui uma base de conhecimento crescente. O **Experimento 2** tem comportamento semelhante se comparado ao **Experimento 1**. À medida que o valor do *suporte mínimo* é reduzido, o número de regras geradas aumenta. A maioria das regras de associação extraídas também estavam no **Experimento 1**. Ambos os métodos para redução de regras diminuíram em 33.60% a quantidade de regras em comparação com o algoritmo original.

Ao analisar o algoritmo original, foram geradas 20 regras a mais do que no **Experimento 1** para o *suporte mínimo* de 0.01. Porém, ao se observar a quantidade de regras extraídas com a aplicação dos métodos de eliminação de regras no **Experimento 2**, percebe-se uma redução de mais de 10% (40 regras) devido ao método de eliminação de regras irrelevantes, principalmente.

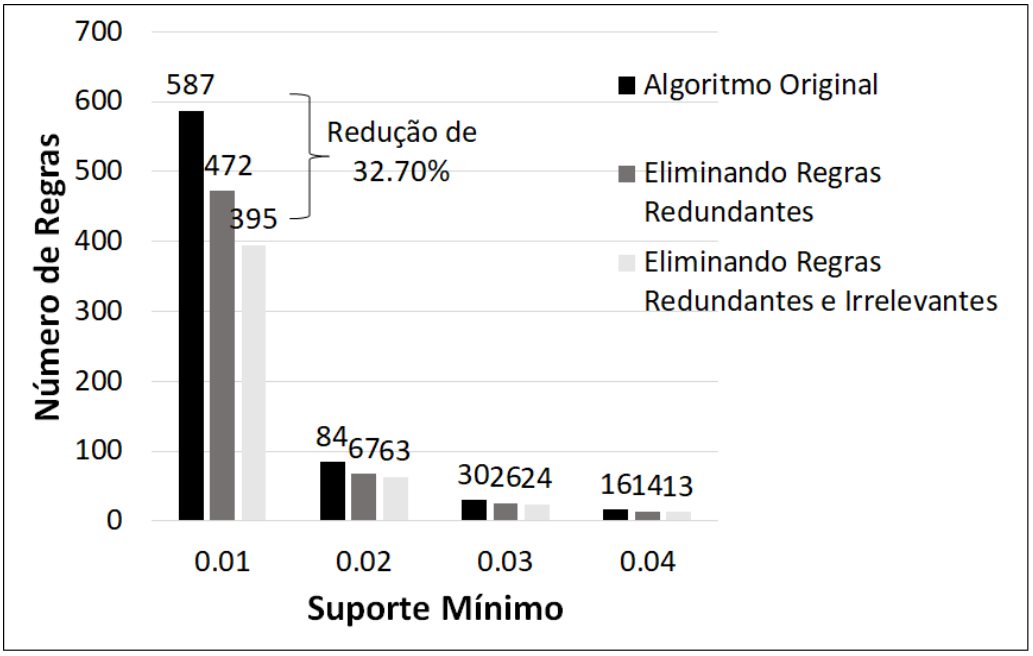

Figura 7.8: Número de Regras Extraídas pelo Experimento 3

A razão para esse comportamento está nas regras de associação irrelevantes geradas na última iteração do **Experimento 1** (com *suporte mínimo de 0.01*). Essas só podem ser utilizadas para eliminar super CR irrelevantes em futuras iterações do algoritmo. Por exemplo, considere que as seguintes regras irrelevantes tenham sido geradas na última iteração do **Experimento 1**:

1: *atletaTemTécnico(X, tony_la_russa)* $\rightarrow$ *atletaJogaEstadio(X, busch_stadium)*

2: *atletaTemTécnico(X, tony_la_russa)* $\rightarrow$ *atletaJogaEstadio(X, busch_stadium), atletaJogaLiga(X, mlb).*

As regras 1 e 2 são úteis em futuros ciclos de iteração, e suas *super CRs* não serão geradas no futuro. De fato, no **Experimento 2**, a segunda regra não foi gerada, uma vez que é uma *super CR* da regra de associação irrelevante 1. Isso explica a redução de mais de 30% nas regras de associação com *suporte mínimo* de 0.01 em comparação com o algoritmo original, e porque os métodos de redução geraram menos regras no **Experimento 2** se comparado ao **Experimento 1**.

O **Experimento 3** é ilustrado na figura 7.8. O conjunto de dados utilizado também foi incrementado em, aproximadamente, 10%. Ambos métodos extraíram, novamente, menos regras de associação (32.70% menos) que o algoritmo original. Portanto, esses experimentos demonstraram o quão importante e necessário os métodos de redução de regras introduzidos

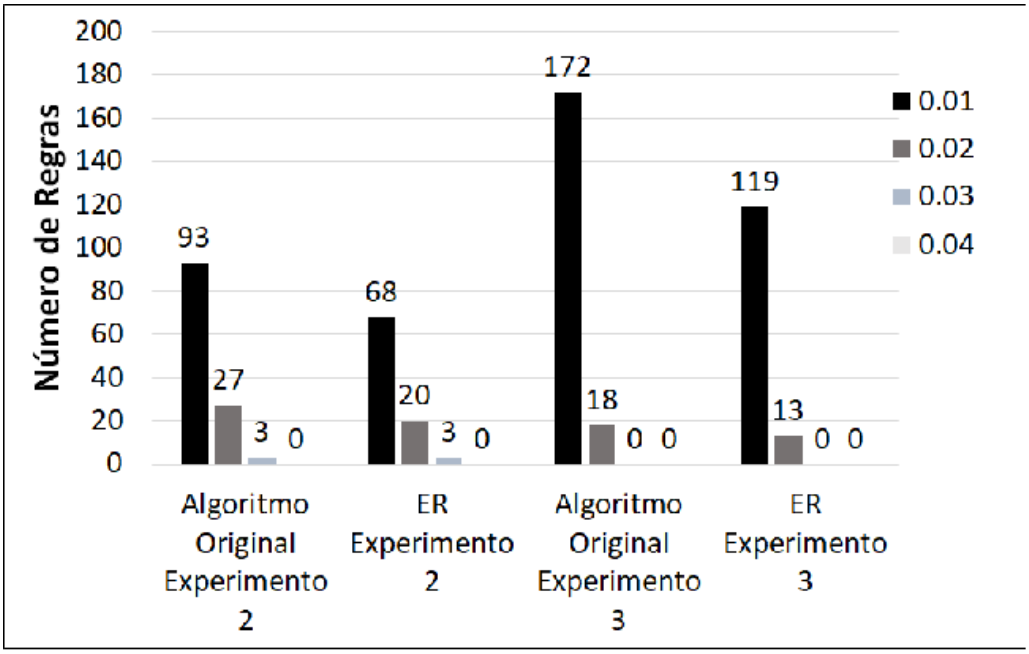

Figura 7.9: Novas Regras obtidas nos Experimentos 2 e 3

neste trabalho são para diminuir o número de regras de associação redundantes e irrelevantes, sem nenhuma perda no processo de popular uma grande base de conhecimento crescente.

No entanto, se for feita uma comparação entre o número de regras obtidas pelos métodos de redução de regras nos **Experimentos 2** e **3**, é possível notar que o último extraiu uma quantidade maior de regras. A explicação para isso está nos tipos de fatos que foram adicionados ao conjunto de dados em cada ciclo de iteração. A figura 7.9 mostra somente as novas regras extraídas pelo algoritmo original e pelos métodos de eliminação de regras nos **Experimentos 2** e **3**, e a tabela 7.2 traz algumas regras de associação extraídas em cada experimento.

O conjunto de dados usado nos **Experimentos 1** e **2** tinha fatos relacionados a esportes, principalmente para *basquete*, *futebol americano* e *baseball*, por exemplo. Algumas novas regras foram descobertas no **Experimento 2** devido à adição de novos fatos, como novos times, principalmente. Por exemplo, as regras 3 e 4 da tabela 7.2 somente aparecem no **Experimento 2**, possivelmente pela adição de fatos relacionados a atletas que jogam pelas equipes *spurs* e *red_sox*.

No **Experimento 3**, mais fatos relacionados a outros esportes (*futebol*, por exemplo) devem ter sido adicionados, resultando em mais regras relevantes e irrelevantes, explicando a razão pela qual os métodos de eliminação de regras trouxeram mais regras de associação com o valor de *suporte mínimo* de 0.01.

Entretanto, ao comparar a quantidade de regras geradas pelo algoritmo original com as

Tabela 7.2: Exemplos de Regras Extraídas em Cada Experimento

Número	Regra	Experimento
1	*atletaGanhaTroféu(X, super_bowl)* → *atletaJogaEsporte(X, Futebol Americano)*	1
2	*atletaJogaLiga(X, nba)* → *atletaJogaEsporte(X, Basquete)*	1
3	*atletaJogaTime(X, spurs)* → *atletaJogaLiga(X, nba)*	2
4	*atletaJogaTime(X, red_sox)* → *atletaJogaLiga(X, mlb)*	2
5	*atletaJogaTime(X, real_madrid)* → *atletaJogaEsporte(X, Futebol)*	3
6	*atletaJogaLiga(X, liga_campeões)* → *atletaJogaEsporte(X, Futebol)*	3

regras obtidas após a aplicação do componente ER, as regras de associação redundantes e irrelevantes foram eliminadas pelo ER, principalmente as com valor de *suporte mínimo* de 0.01, demonstrando a importância e a eficiência dessas técnicas.

Além de comparar os métodos de redução de regras com o algoritmo original, duas outras comparações foram feitas com outros algoritmos. A primeira diz respeito a quantidade de itemsets frequente obtidos pelo método proposto neste trabalho após a aplicação do parâmetro MSC com a quantidade obtida pelos algoritmos CHARM (para mineração de FCIs), FPMax (MFIs), e com um algoritmo tradicional para mineração de itemsets frequentes, o FP-Growth. A segunda verifica a porcentagem de valores ausentes que o ER, o CHARM e o FP-Growth preencheram com fatos o subconjunto de dados, com o intuito de verificar qual algoritmo é mais eficiente nesse processo.

A figura 7.10 traz a quantidade de itemsets frequentes descobertos pelo algoritmo proposto neste trabalho (para mineração de FIMVs, utilizando o parâmetro MSC), pelo algoritmo FP-Growth (FIs), CHARM (FCIs) e FPMax (MFIs) no **Experimento 1**. Como descrito no capítulo 2, existe o seguinte relacionamento: MFI $\subseteq$ FCI $\subseteq$ FI $\subseteq$ FIMV. Ele pode ser observado por meio da quantidade de itemsets frequentes trazidos na figura 7.10. Um algoritmo tradicional de mineração de itemsets frequentes obtém pelo menos a mesma quantidade de itemsets que algoritmos para a extração de itemsets fechados e maximais. Logo, o FP-Growth tem maior probabilidade de obter mais itemsets frequentes do que o CHARM e o FPMax. Isso também irá refletir na quantidade de valores ausentes que o FP-Growth pode preencher, conforme a tabela 7.3.

No entanto, ao se comparar o algoritmo proposto neste trabalho (que utiliza o parâmetro MSC durante a geração de candidatos) com os três algoritmos, um número bem maior de itemsets frequentes é gerado. Portanto, o algoritmo de mineração de regras de associação deste trabalho irá gerar mais regras (que podem ser úteis para popular a BC do NELL) do que o FP-Growth, o CHARM e o FPMax. Caso a BC não fosse composta por valores ausentes, a quantidade de itemsets frequentes obtidas pelo algoritmo deste trabalho e por um algoritmo tradicional seria a mesma. Contudo, muitas dessas regras são redundantes e irrelevantes, fazendo

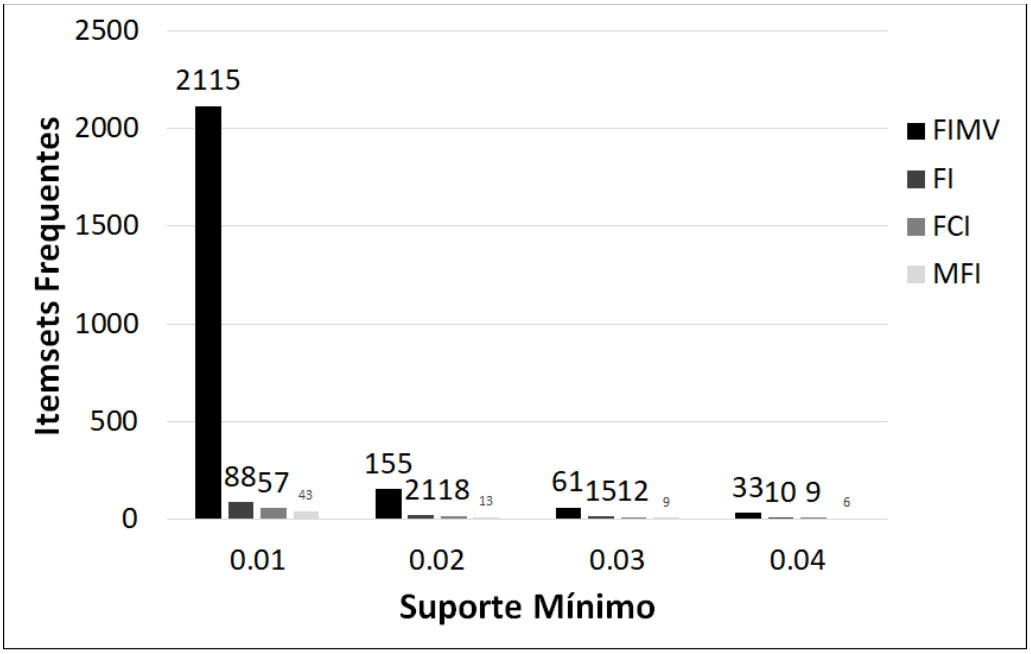

Figura 7.10: Quantidade de FIMVs, FIs, FCIs e MFIs

que o ER seja necessário para reduzir o esforço ao avaliar as regras.

Os algoritmos CHARM e FPMax eliminam itemsets redundantes, gerando somente itemsets frequentes fechados e maximais, respectivamente. No entanto, eles podem eliminar alguns itemsets importantes para popular grandes BCs incompletas. Dessa forma, um algoritmo tradicional é mais eficiente para popular grandes BCs. Como já mencionado, o descarte de itemsets pelo parâmetro MSC resulta em itemsets com suportes maiores que, consequentemente, geram mais itemsets frequentes. Assim, o algoritmo proposto neste trabalho gera mais itemsets frequentes, o que gera mais regras relevantes para popular a BC do NELL.

A tabela 7.3 mostra a porcentagem de valores ausentes preenchidos após a aplicação do componente ER no **Experimento 1** em comparação com o FP-Growth e o CHARM. A tabela 7.4 traz algumas regras de associação extraídas pelo ER, FP-Growth, CHARM, e algumas possíveis regras de associação geradas a partir dos MFIs do algoritmo FPMax. Vale salientar que são utilizadas as implementações dos algoritmos FP-Growth e CHARM feitas por (FOURNIER-VIGER et al., 2014) e que não é conhecida, até o momento, nenhuma implementação de um algoritmo de regras de associação com base em MFIs. A principal razão para esse fato é que não há uma maneira rápida de se obter o suporte dos antecedentes/consequentes de uma regra de associação sem uma nova varredura no conjunto de dados.

Como pode ser observado na tabela 7.3, até mesmo com o valor de *suporte mínimo* de 0.01, o algoritmo CHARM não preencheu a mesma quantidade de valores ausentes que o ER fez com *mínimo suporte* de 0.04. Isso mostra que o uso das regras de associação minimais, em vez de

Tabela 7.3: Porcentagem de valores ausentes preenchidos no Experimento 1

Suporte Mínimo	ER	FP-Growth	CHARM
0.01	5.86%	3.09%	2.89%
0.02	5.58%	2.09%	2.09%
0.03	5.18%	1.4%	1.4%
0.04	3%	1.4%	1.4%

Tabela 7.4: Exemplo de Regras extraídas por ER, FP-Growth, CHARM e FPMax

Número	Regras de Associação	Algoritmo
1	*atletaGanhaTroféu(X, campeonato_nba), atletaJogaLiga(X, nba) → atletaJogaEsporte(X, Basquete)*	FP-Growth / CHARM / FPMax
2	*atletaJogaLiga(X, nba) → atletaJogaEsporte(X, Basquete)*	ER / FP-Growth / CHARM
3	*atletaGanhaTroféu(X, campeonato_nba) → atletaJogaEsporte(X, Basquete)*	ER / FP-Growth
4	*atletaJogaTime(X, cleveland_cavalier), atletaJogaLiga(X, nba) → atletaJogaEsporte(X, Basquete)*	FP-Growth / CHARM
5	*atletaJogaTime(X, cleveland_cavalier) → atletaJogaEsporte(X, Basquete)*	ER / FP-Growth
6	*atletaJogaTime(X, boston_celtics) → atletaJogaEsporte(X, Basquete)*	ER
7	*atletaJogaEstadio(X, amway_arena) → atletaJogaEsporte(X, Basquete)*	ER

as eliminar, auxilia a popular grandes BCs com uma maior quantidade de valores. Além disso, ao comparar o FP-Growth com o CHARM, tem-se que o algoritmo FP-Growth preenche mais valores ausentes do que o CHARM apenas com o suporte de 0.01. A razão para tal é que para os *suportes mínimos* de 0.02, 0.03 e 0.04 apenas regras com dois itemsets foram geradas por esses algoritmos. Desse modo, não é eliminado nenhum itemset pelo CHARM para gerar regras de associação, mas apenas itemsets de tamanho 1. O mesmo não ocorre com o *suporte mínimo* de 0.01, com o qual o FP-Growth utiliza as regras minimais e as super-regras para popular a BC, enquanto o CHARM utiliza apenas as regras obtidas a partir dos FCIs. Isso mostra, novamente, que regras minimais são mais úteis no processo de popular grandes BCs.

Comparando o FP-Growth com o algoritmo proposto neste trabalho, alguns fatos podem ser observados. Em primeiro lugar, as regras geradas após a aplicação do ER auxiliam a popular a BC com mais fatos que o FP-Growth. Isso não se deve aos métodos de redução de regras do ER, mas sim ao parâmetro MSC. Em segundo lugar, com a eliminação das super ARs e super CRs pelo ER, apenas as regras minimais necessárias para preencher a BC são utilizadas. O FP-Growth gera tanto as regras minimais quanto as super-regras, o que torna a avaliação das mesmas um esforço desnecessário.

Ao se observar a tabela 7.4, é possível notar que algumas regras de associação foram extraídas somente pelo método proposto neste trabalho de doutorado (regras 6 e 7), resultado do parâmetro MSC. Outras foram geradas somente pelo FP-Growth e CHARM (1 e 4), uma vez que o ER as elimina, já que o ER gera suas duas sub ARs (regras 2 e 3 são as minimais da regra 1, e as regras 2 e 5 são as minimais da regra 4). Pelas regras 1 e 4, o FP-Growth e CHARM somente podem popular uma BC se ambos os antecedentes estiverem presentes na instância. No caso do FP-Growth, ele também gerou as duas sub ARs para a regra 1, o que não prejudica

na população da BC, mas traz uma regra redundante a mais para ser analisada (regra 1). Nesse caso, se apenas o valor de *nba* ou de *campeonato_nba* estiverem presentes na instância, a BC será atualizada com o valor *basquete*.

Em uma simples comparação com o FPMax, considere-se o itemset frequente (*nba, basquete, campeonato_nba*) que foi gerado com o *suporte mínimo* de 0.01. Nenhum de seus subconjuntos foi gerado pelo FPMax. A regra 1 da tabela 7.4 é uma possível regra de associação que poderia ser gerada a partir desse MFI. Ela só poderá atualizar uma BC se ambos os antecedentes aparecerem em uma instância. Logo, o algoritmo proposto neste trabalho é mais eficiente para eliminar regras de associação redundantes e irrelevantes em grandes bases de conhecimento, também contribuindo para populá-las de maneira mais eficaz do que os algoritmos FP-Growth, CHARM e FPMax.

7.2 Experimentos TARE e TCI

Os experimentos dessa seção tem como finalidade avaliar os componentes TARE e TCI nos seguintes aspectos:

- Quantidade de STARs geradas pelo TARE;
- Número de possíveis inconsistências descobertas pelas métricas probabilísticas (TARE) e pelo método de detecção de inconsistência (TCI);
- Quantidade de atualizações feitas na base de conhecimento pelo TCI;
- Modo como as correções são realizadas.

Para realizar os experimentos, os mesmos valores para o *suporte mínimo* e a *confiança mínima* da seção 7.1 foram utilizados - o suporte variando de 0.04 para 0.01 e decrescendo em 0.01 após cada iteração do ciclo; a confiança mínima fixada em 0.3. Os subconjuntos de dados utilizados também são os mesmos, tendo dados esportivos extraídos da BC do NELL, conforme a figura 7.1.

Os subconjuntos utilizados foram os mesmos da fase 2 da seção 7.1, ou seja, foram utilizados três subconjuntos. Em cada iteração do ciclo, o subconjunto é atualizado pelas regras de associação relevantes, pelo componente TCI e pelas correções das inconsistências encontradas. Ao final de cada ciclo de iteração, o subconjunto é atualizado com a adição de novos fatos da BC do NELL. Lembrando que cada ciclo de iteração possui 4 iterações com os diferentes valores para o *suporte mínimo*. Assim, têm-se os mesmos experimentos:

- **Experimento 1**: subconjunto de dados 1 e ciclo 1;
- **Experimento 2**: subconjunto de dados 2 e ciclo 2;
- **Experimento 3**: subconjunto de dados 3 e ciclo 3.

Para realizar os experimentos, dois métodos foram testados com o subconjunto de dados do **Experimento 1**. Em um método, foi realizado um estudo para identificar quais categorias do subconjunto teriam uma característica *pontual*. No outro, as métricas probabilísticas do TARE foram utilizadas para identificar as pontualidades das categorias. Foi identificado que, ao fixar quais domínios são *pontuais* e quais não o são no primeiro ciclo de iterações, as regras de associação não possuem erros com relação às suas pontualidades, isto é, uma regra que deveria ser *pontual* não foi classificada erroneamente como *não pontual*. Por outro lado, quando não é realizado esse estudo, algumas regras de associação são classificadas com erros relacionados às pontualidades, o que pode afetar em atualizações com erro no componente TCI, contribuindo com a propagação de erros.

Na figura 7.11 pode ser observado que o número de regras de associação com informação incorreta com relação ao tipo de sua pontualidade (ser ou não *pontual*) é maior quando os domínios não são fixados. Se o algoritmo não souber a priori quais são os domínios do subconjunto que possuem característica *pontual*, ele pode erroneamente classificar uma regra como *pontual* ou *não pontual*. Portanto, é importante identificar quais domínios possuem a característica *pontual* no primeiro ciclo de iteração, com o intuito de classificar corretamente a pontualidade de uma regra, o que irá ajudar a detectar melhor as possíveis inconsistências e realizar a atualização da base de dados sem erros. Para a estrutura do subconjunto utilizado, a categoria *Troféu* foi identificada como sendo pontual. Portanto, as regras de associação que possuírem esse domínio serão consideradas pontuais no primeiro ciclo de iterações. A partir do segundo ciclo, as métricas probabilísticas são responsáveis por classificar uma regra como pontual ou não.

A figura 7.12 traz o resultado do **Experimento 1**, em que são exibidas as quantidades de STARs, as possíveis inconsistências encontradas pelo TARE e pelo TCI, e a quantidade de correlações temporais encontradas para atualizar a base de conhecimento do NELL pelo componente TCI.

Pode ser observado que o número de STARs aumenta conforme o *suporte mínimo* desejado diminui. Isso se deve a dois principais motivos: (i) devido à maior quantidade de regras geradas com suportes menores, o que resulta em mais STARs, e (ii) devido às inconsistências detectadas, que são corrigidas e adicionadas na tabela pontual quando necessário. Também é possível

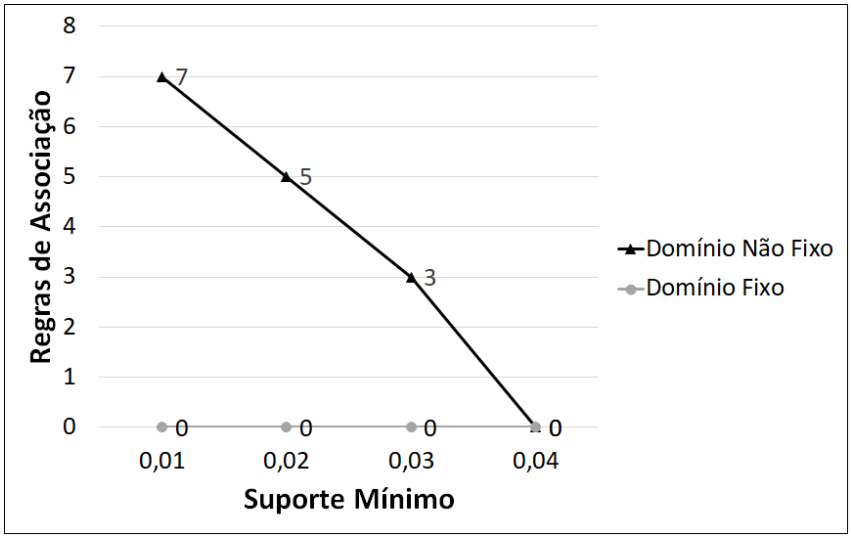

Figura 7.11: Número de Regras de Associação com erros de pontualidade

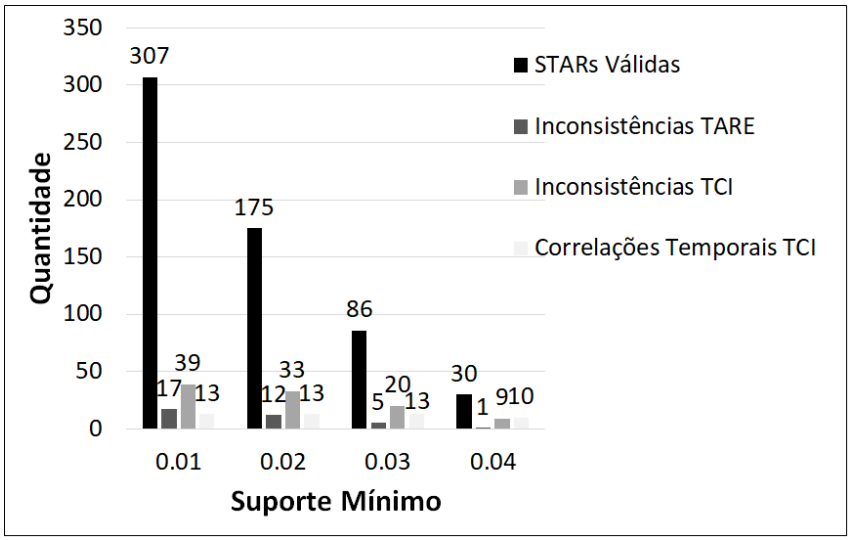

Figura 7.12: Quantidade de STARs, possíveis inconsistências pelo TARE e TCI, e Correlações Temporais

observar que o número de correlações temporais ou cresce ou continua igual ao do suporte anterior. As correlações temporais, somadas às regras de associação auxiliam a popular a BC do NELL, diminuindo a quantidade de valores ausentes.

Como se pode observar na figura 7.12, a quantidade de possíveis inconsistências detectadas também aumenta com a diminuição do *suporte mínimo*. No entanto, o componente TCI encontrou mais possíveis inconsistências se comparado ao TARE, inclusive a maioria das que o TARE encontrou. A figura 7.13 mostra a quantidade de inconsistências reais encontradas por ambos os componentes no **Experimento 1**. A tabela 7.5 traz algumas instâncias inconsistentes, e a tabela 7.6 demonstra como elas são corrigidas na tabela pontual.

É possível notar, na figura 7.13, que o número de inconsistências reais é um pouco menor

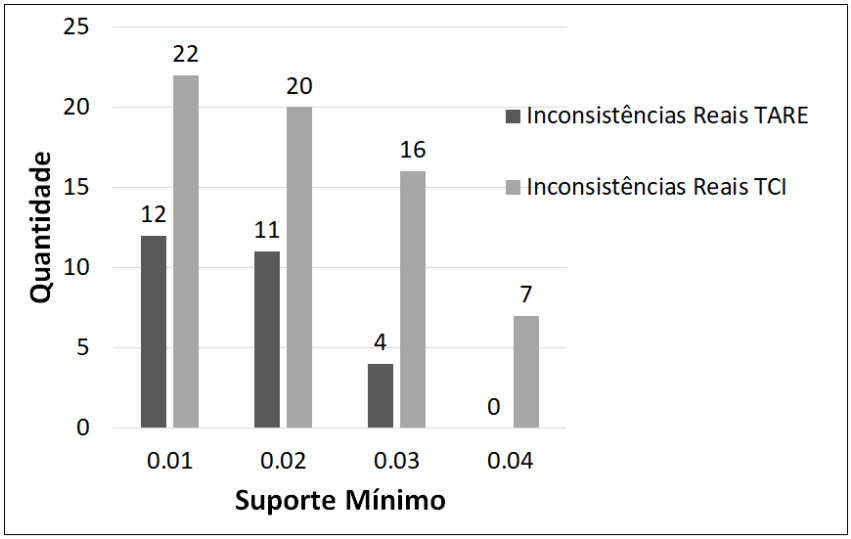

Figura 7.13: Quantidade de inconsistências reais TARE x TCI no Experimento 1

Tabela 7.5: Instâncias inconsistentes na base de conhecimento

Atleta	Técnico	Esporte	Liga Esportiva	Troféu	Time	data inicial	data final
Ben Roeth-lisberger	Mike Tomlin	Futebol Americano	nfl	super bowl	Pittsburgh Steelers	2004	2014
Eli Manning	Tom Coughlin	Futebol Americano	nfl	super bowl	Giants	2004	2014
Larry Foote	mv	mv	mv	super bowl	Pittsburgh Steelers	2002	2008
Tom Brady	mv	mv	mv	super bowl	New England Patriots	2000	2014

que o de possíveis inconsistências e que o componente TCI encontra mais inconsistências reais. Isso se deve ao fato de o componente TCI utilizar toda STAR válida gerada e tentar encontrar correlações temporais com todo o subconjunto de dados. Já no TARE, são utilizadas apenas as pontualidades das regras de associação extraídas.

Considerando a tabela 7.5, a qual possui amostras de instâncias detectadas como inconsistentes, tem-se que, durante a primeira iteração do **Experimento 1**, a STAR envolvendo o atleta *Ben Roethlisberger* foi gerada, com suas datas inicial e final 2004 e 2014, respectivamente. O TCI procura por correlações temporais com alguma intersecção temporal e, com base na adaptação de Allen proposta neste trabalho, verifica quais são outras categorias válidas para realizar a correlação (nesse caso *Técnico*, *Troféu* e *Time*), e (i) realiza atualizações quando são encontradas instâncias relacionadas ou (ii) detecta uma possível inconsistência em uma ou mais instâncias. Note-se que as linhas 2 e 4 da tabela são instâncias retornadas por ter um valor em comum para a categoria *Troféu*. Existe alguma intersecção entre os intervalos temporais (*equal* e *finishes*, respectivamente), mas alguns outros valores são diferentes, como o *Time*, por exemplo. Isso caracteriza uma possível inconsistência.

Ao analisar cada possível inconsistência, detectou-se que existe informação incorreta com

Tabela 7.6: Instâncias na Tabela Pontual Corrigidas

Atleta	Técnico	Esporte	Liga Esportiva	Troféu	Time	data inicial	data final
Ben Roethlisberger	Mike Tomlin	Futebol Americano	nfl	super bowl	Pittsburgh Steelers	2005	2005
Ben Roethlisberger	Mike Tomlin	Futebol Americano	nfl	super bowl	Pittsburgh Steelers	2008	2008
Eli Manning	Tom Coughlin	Futebol Americano	nfl	super bowl	Giants	2007	2007
Eli Manning	Tom Coughlin	Futebol Americano	nfl	super bowl	Giants	2011	2011
Larry Foote	Mike Tomlin	Futebol Americano	nfl	super bowl	Pittsburgh Steelers	2002	2008
Larry Foote	Mike Tomlin	Futebol Americano	nfl	super bowl	Pittsburgh Steelers	2002	2008
Tom Brady	mv	mv	mv	super bowl	New England Patriots	2000	2014
Tom Brady	mv	mv	mv	super bowl	New England Patriots	2000	2014
Tom Brady	mv	mv	mv	super bowl	New England Patriots	2000	2014
Tom Brady	mv	mv	mv	super bowl	New England Patriots	2000	2014

relação à categoria *Troféu*, a qual possui característica pontual, mas nas instâncias em questão o intervalo é não pontual. Por exemplo, é possível que o atleta *Tom Brady* tenha vencido o torneio *super bowl* em um ou alguns anos entre a *data inicial*(2000) e *final*(2014), mas não em todos os anos.

Na tabela 7.6, as inconsistências são corrigidas. É possível observar que todos os atletas da tabela 7.5 venceram o *Troféu super bowl* mais de uma vez. Assim, uma instância é adicionada à tabela pontual para cada vez que o atleta venceu determinado campeonato (característica pontual). Isso evita dois tipos de problemas:

- Tornar *pontuais* todas as categorias do conjunto de dados;
- Aumentar o suporte dos itemsets.

Para o primeiro item, se todas as instâncias corrigidas fossem adicionadas na tabela original de dados, domínios com característica não pontual, como *Time* e *Liga Esportiva* (que representa as relações de um atleta jogando por um time e em uma liga), tornariam-se pontuais. A mesma adição na tabela original faria com que alguns itemsets tivessem maior suporte do que deveriam, como *super bowl, nfl e futebol americano*. Dessa forma, a criação da tabela pontual resolve esses problemas, corrigindo as inconsistências. Note-se que a instância original é mantida. No entanto, essas possíveis inconsistências não serão investigadas novamente, uma vez que já foram analisadas e corrigidas.

Contudo, o componente TCI também trabalha com correlação temporal no intuito de atualizar a BC. As linhas 1 e 3 da tabela 7.5 possuem intersecção temporal (during) e, além do

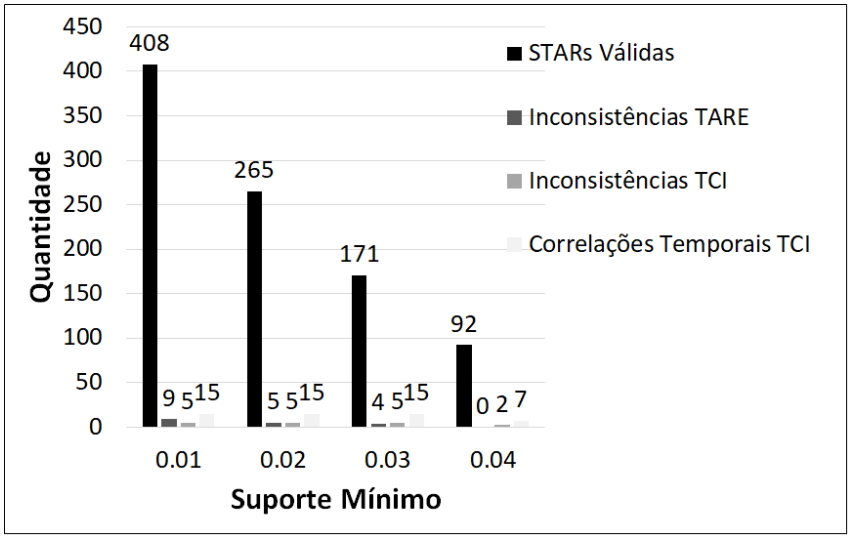

Figura 7.14: Quantidade de STARs, possíveis inconsistências pelo TARE e TCI, e Correlações Temporais - Experimento 2

valor *super bowl*, também têm o time *Pittsburgh Steelers* como outro valor comum em uma categoria relevante para a atualização. Ao observar os dados do atleta *Larry Foote* na tabela 7.6, identifica-se que alguns valores ausentes foram substituídos, como os valores para técnico, esporte e liga esportiva. Não existia regra de associação para realizar tal preenchimento de dados. Logo, foi a correlação temporal do TCI que efetuou essa atualização, contribuindo para diminuir a quantidade de valores ausentes.

Os resultados do **Experimento 2** são mostrados na figura 7.14. Ao se compararem esses dados com os da figura 7.12, que traz os dados do **Experimento 1**, é possível observar dois principais pontos:

- Aumento do número de STARs;
- Diminuição na quantidade de possíveis inconsistências.

O aumento do número de STARs se deve a dois motivos. Primeiro, a correção de inconsistências, que resulta no aumento de instâncias na tabela pontual, e o aumento da base de dados com a inserção de novos dados. Por outro lado, o número de possíveis inconsistências diminui. Esse número poderia aumentar dependendo dos dados e da quantidade de dados adicionados. A diminuição de possíveis inconsistências ocorreu, principalmente, em virtude de sua correção. Logo, somente novas inconsistências apareceram, todas relacionadas aos novos fatos adicionados à base de conhecimento.

De modo similar ao que ocorre no **Experimento 1**, o número de STARs, de possíveis inconsistências e de correlações temporais aumentam com a diminuição do *suporte mínimo*.

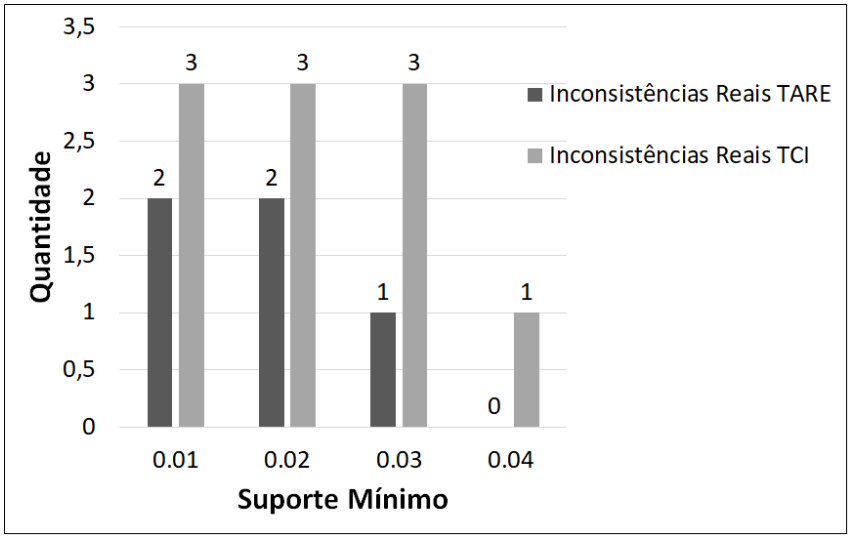

Figura 7.15: Quantidade de inconsistências reais TARE x TCI no Experimento 2

A figura 7.15 mostra a quantidade de inconsistências reais obtidas pelo TARE e pelo TCI para o **Experimento 2**. A quantidade de possíveis inconsistências diminuiu, da mesma forma que a de inconsistências reais. Algumas possíveis inconsistências não são reais por dois principais fatos:

- Métricas probabilísticas sugerirem uma possível inconsistência pela pontualidade de uma regra não pontual (TARE);
- Um mesmo evento pontual ocorrer mais de uma vez no mesmo ano (TCI).

Para uma melhor compreensão desses dois problemas, observe-se a tabela 7.7, que traz exemplos de algumas instâncias do subconjunto de dados. As instâncias relacionadas aos dois primeiros atletas foram caracterizadas como possíveis inconsistências pela métrica probabilística MP1. Suas respectivas STARs foram geradas a partir da regra *AR1* a seguir:

*AR*1: *atletaJogaLiga(X, mlb)* $\rightarrow$ *atletaJogaEsporte(X, Baseball)*

Tal regra possui a característica *não pontual*. No entanto, as respectivas STARs dos atletas envolvidos são pontuais. Contudo, isso não se caracteriza como um erro. Embora a MP1 sugira essas STARs como inconsistentes, elas não o são, uma vez que um atleta pode jogar em uma liga apenas um ano. A métrica MP2 também sugere que são inconsistentes, já que outras regras de associação com os mesmo domínios também possuem característica *não pontual*. Dessa forma, as instâncias das duas primeiras linhas da tabela 7.7 não são inconsistentes.

Tabela 7.7: Instâncias não inconsistentes na base de conhecimento

Atleta	Técnico	Esporte	Liga Esportiva	Troféu	Time	data inicial	data final
Brad Lidge	mv	Baseball	mlb	world series	mv	2008	2008
Curt Schilling	mv	Baseball	mlb	world series	mv	2001	2001
Dinara Safina	mv	mv	mv	French Open	mv	2009	2009
Dinara Safina	mv	mv	mv	Grand Slam	mv	2009	2009
Dinara Safina	mv	Tênis	mv	Australian Open	mv	2009	2009

As instâncias das linhas de 3 a 5 foram identificadas como possíveis inconsistências pelo componente TCI. Nesse exemplo, o algoritmo busca pelos dados relacionados à atleta *Dinara Safina*, que venceu o torneio *Australian Open* no ano de 2009. A STAR correspondente para a busca foi gerada para formar a regra AR2 a seguir:

*AR*2: *atletaGanhaTroféu(X, Australian Open)* $\rightarrow$ *atletaJogaEsporte(X, Tênis).*

O algoritmo busca por dados relacionados à atleta em questão durante o período temporal 2009-2009. O TCI identifica três outras instâncias da mesma atleta para o ano de 2009. Existem três valores distintos para o evento *pontual* que é ganhar um troféu. Porém, isso não caracteriza uma inconsistência pelo fato de (i) a atleta ter vencido tanto o *Australian Open* quanto o *French Open* e (ii) *Grand Slam* ser o tipo dos torneios *Australian Open* e *French Open*. Nesse exemplo, um evento pontual ocorre mais de uma vez no mesmo ano e que o mesmo troféu possui mais de um nome.

Os resultados do **Experimento 3** são ilustrados na figura 7.16. Assim como no **Experimento 2**, o subconjunto utilizado foi incrementado em 10%, aproximadamente, com novos dados da BC do NELL. Foram adicionados dados relacionados a outros esportes - *futebol*, por exemplo. Não foi extraída nenhuma nova regra de associação até o *suporte mínimo* de 0.02. No entanto, podem ser adicionadas diferentes STARs para determinada regra pela adição de novas instâncias que ajudam a gerar a regra.

A quantidade de possíveis inconsistências e correlações temporais também é pequena para os valores 0.04 e 0.03 de *suporte mínimo*, uma vez que não foram geradas regras novas, tendo apenas ocorrido a adição de algumas STARs. Contudo, com os valores de 0.02 e 0.01, novas regras de associação são geradas, trazendo uma quantidade maior de STARs e de possíveis inconsistências e correlações temporais.

Do mesmo modo que o realizado nos dois primeiros experimentos, a figura 7.17 traz a quantidade de inconsistências reais obtidas por cada componente. Aqui, o componente TARE identificou uma inconsistência real a mais que o TCI para o *suporte mínimo* de 0.01. É im-

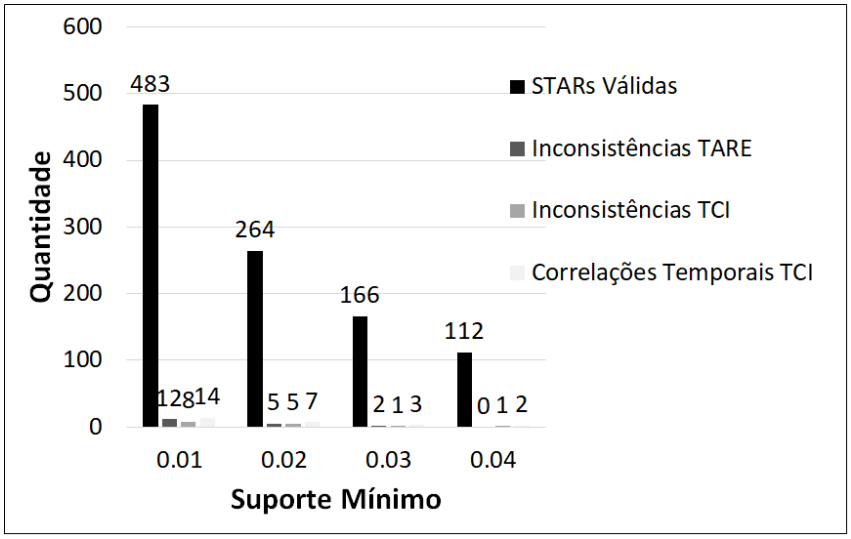

Figura 7.16: Quantidade de STARs, possíveis inconsistências pelo TARE e TCI, e Correlações Temporais - Experimento 3

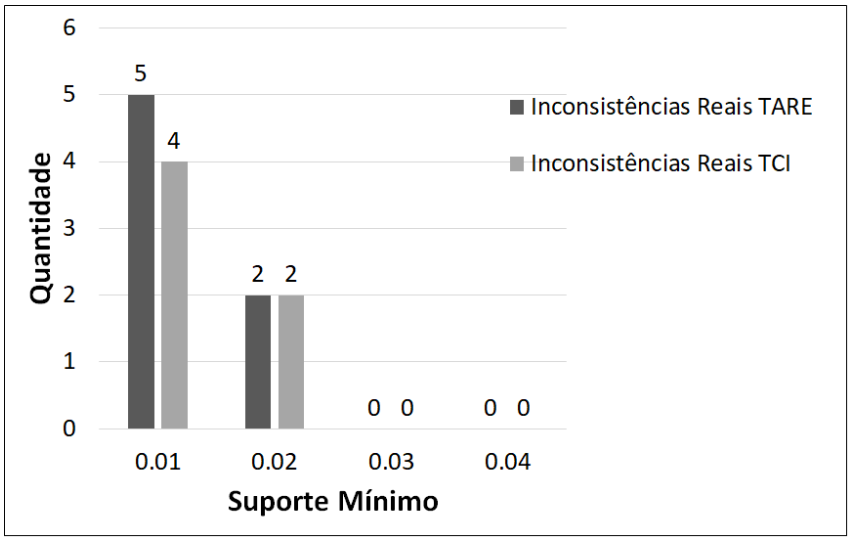

Figura 7.17: Quantidade de inconsistências reais TARE x TCI no Experimento 3

portante notificar que a maioria das inconsistências reais encontradas pelo TARE, também é descoberta pelo TCI. No entanto, o TARE descobre algumas que não são encontradas pelo TCI, como neste caso. Logo, ambos os componentes são necessários para a descoberta de inconsistências.

Para finalizar, a tabela 7.8 traz exemplos de instâncias inconsistentes encontradas nos três experimentos. Todas as inconsistências da tabela estão relacionadas ao domínio *Troféu* (que possui a característica pontual). Todos os intervalos em questão são não pontuais, o que resulta na inconsistência.

Antes do primeiro ciclo de iteração foi realizado um estudo, identificando o domínio *Troféu*

Tabela 7.8: Instâncias inconsistentes na base de conhecimento para os três experimentos

Atleta	Técnico	Esporte	Liga Esportiva	Troféu	Time	data inicial	data final	Experimento
Manu Ginobili	mv	mv	nba	campeonato nba	Spurs	2002	2014	1
Pau Gasol	Phil Jackson	Basquete	nba	campeonato nba	La Lakers	2008	2014	1
Duncan	Greg Popovich	mv	nba	campeonato nba	Spurs	1997	2015	2
Roger Clemans	mv	Baseball	mlb	world series	Yankees	1999	2003	2
Thomas Muller	mv	mv	Liga dos Campeões	Liga dos Campeões da Uefa	Bayern Munique	2008	2015	3
Craig Adams	mv	Hóquei	nhl	stanley cup	Penguins	2009	2014	3

como sendo pontual. Assim, no **Experimento 1**, as instâncias da tabela são apontadas como inconsistentes por ambos os componentes. A partir do segundo ciclo de iterações (neste caso, a partir do **Experimento 2**), as métricas probabilísticas efetuam o trabalho de identificar uma regra de associação como pontual ou não. Logo, as instâncias inconsistentes descobertas nos **Experimentos 2** e **3** levam em consideração a pontualidade de uma regra pelas métricas probabilísticas (TARE) e pela pontualidade do domínio (TCI).

7.3 Análise de Desempenho

Esta seção tem o intuito de mostrar o tempo de execução das principais técnicas abordadas neste trabalho. Embora, o objetivo do trabalho seja popular e detectar inconsistências em GBCs, foi realizado um estudo com relação ao tempo gasto nos seguintes itens:

1. Obter itemsets frequentes;
2. Gerar as regras a partir dos itemsets frequentes;
3. Eliminar regras redundantes (ER);
4. Eliminar regras irrelevantes (ER);
5. Gerar STARs e as métricas probabilísticas (TARE);
6. Realizar a correlação temporal e detecção de inconsistências (TCI).

A análise foi dividida em duas partes. Na primeira, são analisados os tempos de execução dos 4 (quatro) primeiros itens acima, ou seja, o tempo de cada um dos itens para a geração do conjunto final de regras. Na segunda, é realizada uma comparação entre os componentes TARE e TCI.

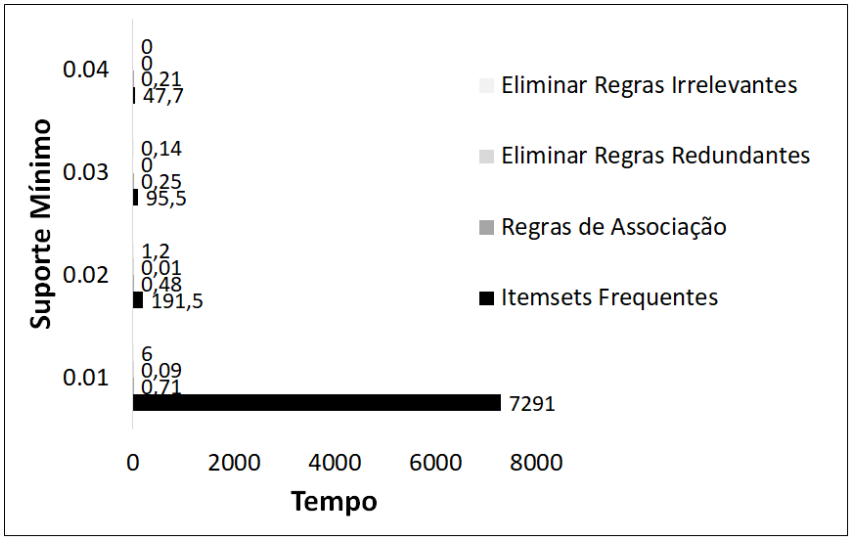

Figura 7.18: Tempo gasto para obter itemsets frequentes, gerar regras, eliminar regras redundantes e irrelevantes no Experimento 1

A figura 7.18 ilustra os resultados do tempo gasto no **Experimento 1** para gerar itemsets frequentes, para obter o conjunto de regras sem a aplicação do componente ER, eliminar regras redundantes e irrelevantes. A medida de tempo utilizada foi segundos. Assim como em todos os algoritmos de extração de regras de associação, conforme o valor do *suporte mínimo* diminui, o tempo para a execução aumenta.

O tempo para gerar itemsets frequentes é bem maior do que se comparado aos demais itens. Isso por dois fatos principais: (i) o passo de junção dos itens, e (ii) o cálculo do suporte pelo parâmetro MSC, que faz uma nova consulta para verificar a quantidade de itemsets que são vazios.

Com o conjunto de itemsets frequentes obtido, o tempo para gerar regras é bem menor. Esses valores correspondem ao tempo despendido para gerar as regras após a geração dos itemsets frequentes.

Ao comparar o desempenho dos dois métodos do componente ER, é possível observar que a eliminação de regras redundantes é mais rápida que a de regras irrelevantes para todos os suportes mínimos. O processo de eliminação de regras redundantes utiliza o conjunto de regras que está armazenado em uma lista de matrizes contendo todas as regras. Assim, o método elimina as super ARs redundantes a partir dessa lista, sem necessidade de consulta à base de dados. No entanto, a eliminação de regras irrelevantes consiste em eliminar super CRs irrelevantes a partir de uma sub CR irrelevante já descoberta, que é armazenada em uma tabela na base de dados. Assim, são realizadas consultas à base de dados, o que torna esse processo mais lento.

Na figura 7.19 são exibidos os resultados de performance com relação ao **Experimento**

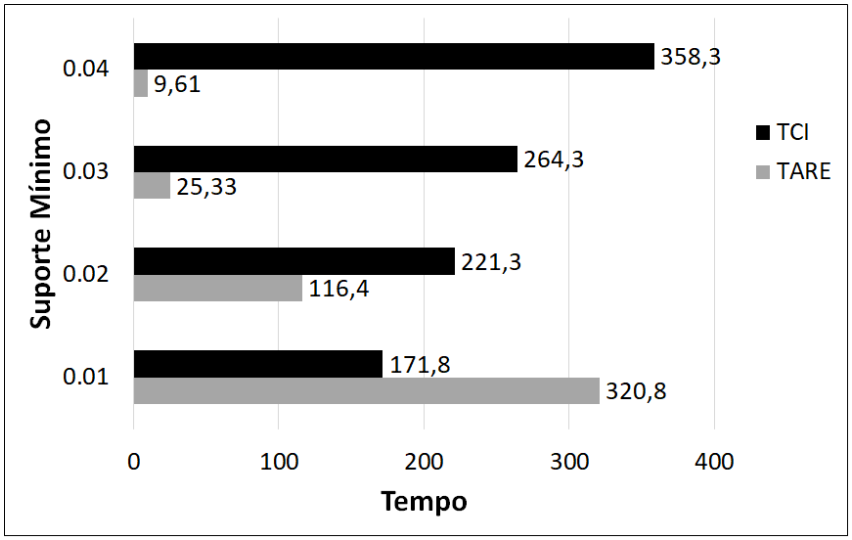

Figura 7.19: Tempo gasto pelo TARE e TCI no Experimento 1

Tabela 7.9: Exemplo de STARs com diferentes suportes.

Número	Regra	Suporte	Data Inicial	Data Final
1	*atletaJogaLiga(ben_roethlisberger, super_bowl)* → *atletaJogaEsporte(ben_roethlisberger, Futebol_Americano)*	0.04	2004	2014
2	*atletaGanhaTrofeu(ben_roethlisberger, super_bowl)* → *atletaJogaEsporte(ben_roethlisberger, Futebol_Americano)*	0.03	2004	2014
3	*atletaJogaTime(ben_roethlisberger, pittsburgh_steelers)* → *atletaJogaEsporte(ben_roethlisberger, Futebol_Americano)*	0.01	2004	2014

1 para os componentes TARE e TCI. O componente TARE realiza a geração de STARs e a verificação das métricas probabilísticas. Cada regra de associação possui um conjunto de STARs. Assim, quanto menos regras geradas, menor é a quantidade de STARs e, consequentemente, menor é o tempo gasto pelas métricas probabilísticas. Como pode ser observado na figura 7.19, conforme o valor do *suporte mínimo* diminui, o TARE leva mais tempo para ser executado.

Contudo, o componente TCI possui um comportamento inverso se comparação ao TARE. Isso se deve a dois fatos principais. Primeiro, a consulta elaborada para realizar a correlação temporal e detecção de inconsistências por uma determinada STAR descoberta com o suporte de 0.04 não será executada novamente com outros *suporte mínimos* no mesmo ciclo (apenas no próximo ciclo de iteração), o que resulta em menos acessos à base de dados para suporte menores.

O segundo motivo se deve ao fato de diferentes STARs estarem associadas à mesma instância. Dessa forma, a mesma consulta seria realizada. Considere as STARs da tabelas 7.9. Nesse caso, tem-se três STARs distintas que envolvem o mesmo atleta e o mesmo período temporal. Ou seja, a mesma instância na base de dados para o atleta ben_roethlisberger com o período temporal

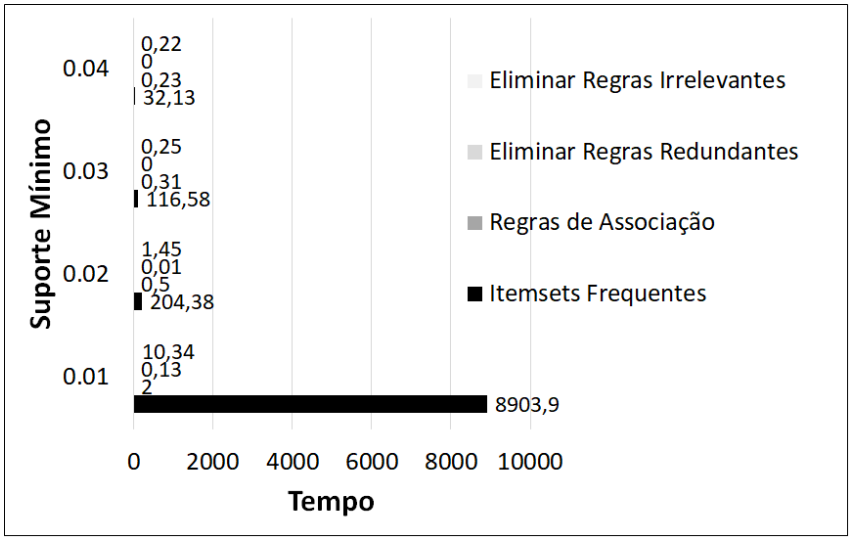

Figura 7.20: Tempo gasto para obter itemsets frequentes, gerar regras, eliminar regras redundantes e irrelevantes no Experimento 2

entre 2004 e 2014 auxilia a gerar três regras diferentes. Assim como descrito na seção 6.3, o TCI verifica quais são seus outros valores válidos para realizar a correlação temporal e realiza a consulta. Como as três regras envolvem a mesma instância, a mesma consulta seria realizada para as três regras. No entanto, a correlação temporal para essa consulta já foi realizada com o *suporte mínimo* de 0.04. Portanto, a mesma não é realizada nas outras iterações do mesmo ciclo. Isso justifica a redução do tempo de execução do TCI a medida que o *suporte mínimo* diminui.

Todavia, em futuros ciclos de iteração, a mesma consulta é realizada novamente (uma vez por ciclo), devido à adição de novas instâncias à base de conhecimento.

Os tempos de execução dos **Experimentos 2** e **3** possuem o mesmo comportamento que o **Experimento 1**. As figuras 7.20 e 7.21 exibem os tempos de execução para obter os itemsets frequentes, gerar as regras de associação, eliminar as regras redundantes e irrelevantes para os **Experimentos 2** e **3**, respectivamente.

A performance para gerar itemsets é um pouco mais lenta se comparado com o **Experimento 1** devido ao aumento da base de dados, o que resulta em uma maior quantidade de itemsets. Um maior número de itemsets pode resultar em uma maior quantidade de regras, o que pode ocasionar mais super ARs redundantes e super CRs irrelevantes. Isso explica o tempo de execução um pouco maior nos **Experimentos 2** e **3**.

O tempo para eliminar as regras redundantes quase não aumenta de um experimento para o outro. Como já relatado, esse método utiliza uma lista de matrizes contendo as regras e as elimina sem necessidade de acesso à base de dados. No entanto, o tempo gasto para eliminar as

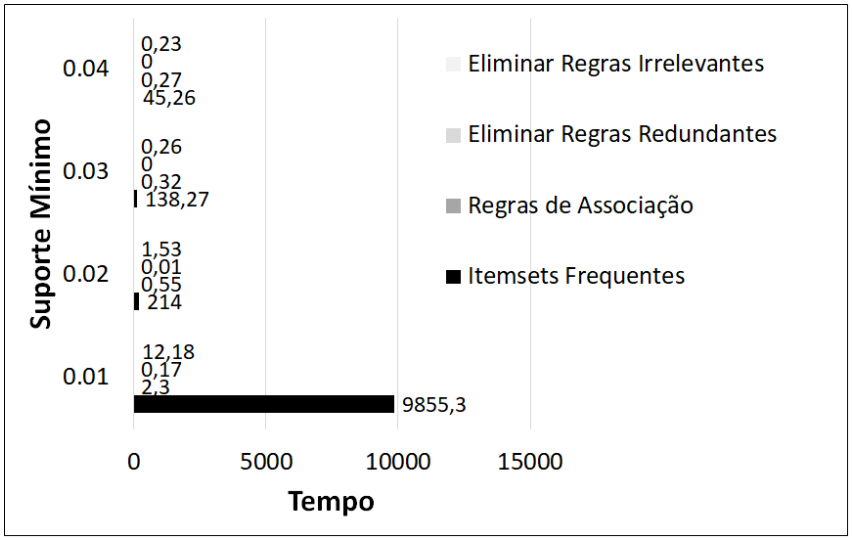

Figura 7.21: Tempo gasto para obter itemsets frequentes, gerar regras, eliminar regras redundantes e irrelevantes no Experimento 3

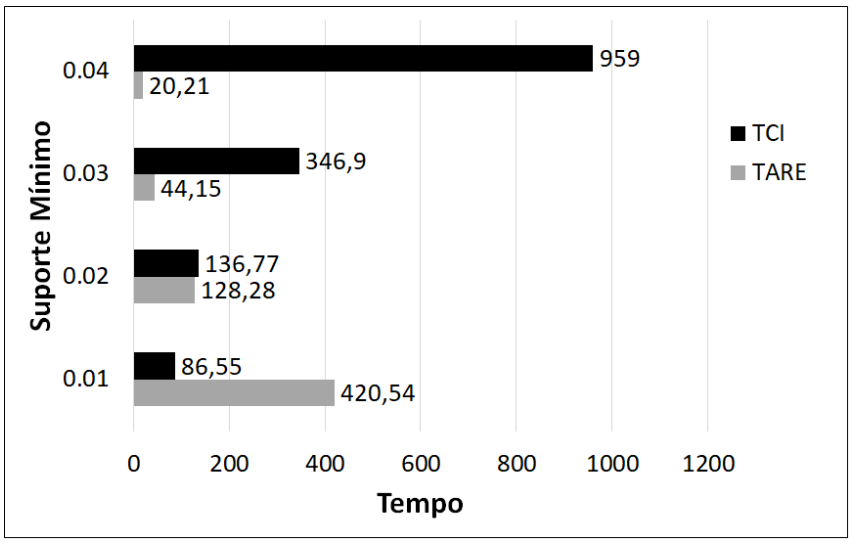

Figura 7.22: Tempo gasto pelo TARE e TCI no Experimento 2

super CRs irrelevantes aumenta um pouco mais a cada experimento, uma vez que consulta as regras irrelevantes que já foram descobertas em iterações anteriores.

Por fim, as figuras 7.22 e 7.23 mostram o tempo gasto para a execução dos componentes TARE e TCI nos **Experimentos 2** e **3**, respectivamente. O comportamento obtido em ambos os experimentos é similar ao ocorrido no **Experimento 1**. Ou seja, para o TARE, conforme o *suporte mínimo* diminui, o tempo de execução para obter as STARs e realizar o cálculo das métricas probabilísticas aumenta. Como o **Experimento 2** possui mais instâncias que o **Experimento 1** (resultando em uma maior quantidade de STARs), o tempo gasto pelo TARE é maior no **Experimento 2**. O mesmo se aplica no **Experimento 3** em comparação com o **Experimento 2**, ou seja, o desempenho do componente TARE é maior no **Experimento 3**.

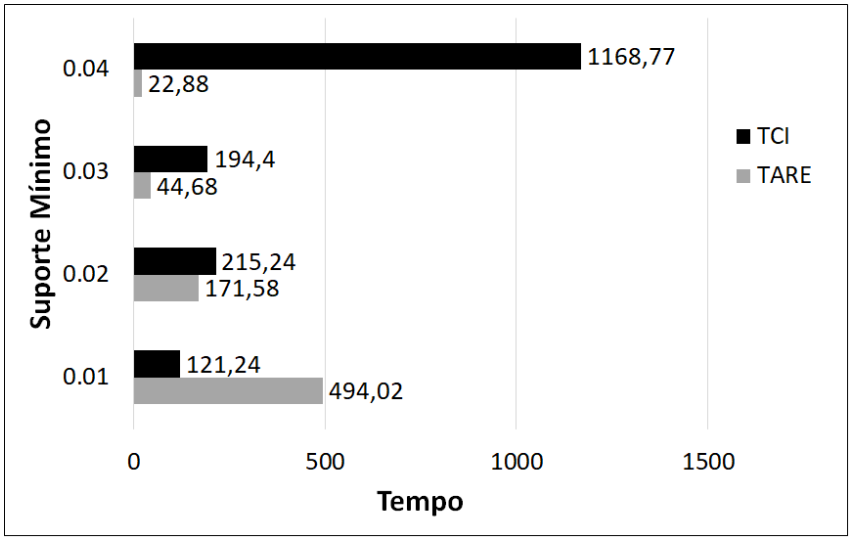

Figura 7.23: Tempo gasto pelo TARE e TCI no Experimento 3

Já para o componente TCI, a medida que o *suporte mínimo* diminui, o tempo de execução também decresce. Exceto no **Experimento 3** entre o *suporte mínimo* de 0.03 e 0.02. Nesse caso, foram utilizadas instâncias que ainda não haviam sido empregadas para gerar regras anteriormente, devido principalmente à descoberta de novas regras de associação que resultam em novas STARs. Assim, consultas inéditas foram criadas, resultando em um tempo de execução maior no TCI para o *suporte mínimo* de 0.02 se comparado com o *suporte mínimo* de 0.03.

Capítulo 8

Conclusão

Este trabalho possui o objetivo de (i) popular a BC do NELL, diminuindo a quantidade de valores ausentes e (ii) detectar inconsistências presentes na mesma. Para tal, foram utilizados a mineração de regras de associação e a correlação temporal.

Grandes bases de conhecimento como a do NELL aumentam seus fatos e relacionamentos diariamente, aplicando algoritmos que buscam por novos dados na Web, ou que desenvolvem técnicas para descobrir relações com base no conhecimento já armazenado. Dessa forma, essas BCs estão longe de serem completas, possuindo muitos valores ausentes.

Para aplicar um algoritmo de extração de regras de associação em uma BC com tal característica é preciso tratar desse problema. Assim, o algoritmo proposto neste trabalho modifica o cálculo tradicional do suporte, criando um novo parâmetro, denominado MSC, que descarta itemsets quando todos os itens de suas categorias estão ausentes. O descarte de itemsets com valores ausentes resulta em itemsets com suporte maiores, o que contribui para gerar mais regras do que os algoritmos tradicionais, populando a BC com mais dados.

Devido ao fato de a BC possuir muitos valores e ser incompleta, muitas regras de associação geradas podem ser consideradas irrelevantes e não devem ser utilizadas para popular a BC, uma vez que podem contribuir para a propagação de erros. Dessa forma, cada regra gerada deve ser analisada, e as regras válidas serão utilizadas para preencher a BC do NELL. Para realizar tal análise, é utilizado o Conversing Learning, um componente do NELL que utiliza usuários do Twitter e do Yahoo Answers para validar regras.

Contudo, a quantidade de regras geradas pode ser muito elevada, e analisar cada regra extraída pode ser um processo exaustivo. Assim, o presente trabalho desenvolveu o componente ER (Eliminating Rules) para eliminar dois tipos de regras de associação no pós-processamento:

1. Regras de Associação Redundantes;
2. Regras de Associação Irrelevantes.

Para o primeiro caso, foi introduzido o conceito de super-regras antecedentes (super ARs). O algoritmo consiste em eliminar super ARs consideradas redundantes, utilizando apenas as sub-regras antecedentes minimais, as quais são mais importantes e eficientes para popular grandes bases de conhecimento.

O conceito de super-regras consequentes (super CRs) foi apresentado para eliminar regras irrelevantes. O processo consiste em eliminar super CRs irrelevantes, com base em uma sub-regra irrelevante descoberta em alguma iteração prévia. Os experimentos mostraram que o componente ER reduz em mais de 30% a quantidade de regras geradas, sem perda no processo de popular grandes BCs.

Portanto, o uso de regras de associação proposto neste trabalho para popular a BC do NELL possui as seguintes contribuições:

- Desenvolvimento do parâmetro MSC para trabalhar com valores ausentes;
- Avaliação de cada regra extraída devido à característica da BC;
- Introdução de super-regras antecedentes para eliminar regras de associação redundantes;
- Introdução de super-regras consequentes para eliminar regras de associação irrelevantes.

Além disso, este trabalho faz uso de temporalidade para popular e detectar inconsistências na BC. Para tal, dois componentes foram desenvolvidos:

- TARE - Temporal Association Rule Extraction;
- TCI - Temporal Correlation Inference.

O TARE introduz as denominadas STARs (Specific Temporal Association Rules) e faz uso de duas métricas probabilísticas com o intuito de detectar inconsistências na base de conhecimento. Uma STAR corresponde a uma regra formada pelos itens específicos utilizados para gerar uma regra de associação, adicionada do intervalo de tempo em que o evento ocorreu. Neste trabalho, utilizam-se os anos inicial e final como a medida de granularidade de tempo.

Uma regra de associação pode ser caracterizada como pontual (se a maioria de suas STARs possuírem a data inicial igual à final) ou como não pontual (data inicial menor que a final). A

partir da característica pontual de uma regra, foram desenvolvidas duas métricas probabilísticas, denominadas MP1 (Métrica Probabilística 1) e MP2 (Métrica Probabilística 2), com o propósito de descobrir inconsistências.

A MP1 verifica se determinada regra de associação é pontual ou não com base nas STARs utilizadas para gerar a regra de associação. Caso uma regra seja pontual, as STARs não pontuais são sugeridas como possíveis inconsistências. Caso contrário, as STARs pontuais serão investigadas.

A MP2 é utilizada para confirmar se determina regra de associação é pontual ou não, verificando todas as regras de associação que envolvem as mesmas categorias nos antecedentes e consequentes da regra. Experimentos demonstraram que as métricas probabilísticas encontram inconsistências, principalmente quando uma regra tem característica pontual e a STAR não é pontual.

O componente TCI é responsável por (i) identificar inconsistências e (ii) obter correlações temporais para povoar a BC do NELL. As técnicas são executadas em paralelo. O TCI recebe como entrada um conjunto válido de STARs e o subconjunto da BC utilizada. Para cada STAR, o algoritmo verifica os demais campos de sua instância e tenta obter correlações temporais. Caso o algoritmo identifique uma outra instância relacionada, ele tenta povoá-la com seus valores (caso esteja com valor ausente). Contudo, se o algoritmo encontra algum conflito envolvendo uma intersecção no período temporal de ambas as instâncias, uma possível inconsistência é identificada. Para realizar as comparações entre os intervalos é feita uma adaptação do intervalo algébrico de Allen (Allen's TIA), em que os intervalos temporais são reduzidos de 13 (Allen's TIA) para 11 na abordagem deste trabalho.

Logo, os componentes TARE e TCI possuem as seguintes contribuições com o objetivo de popular e detectar inconsistências na BC:

- Introdução do conceito de STAR (TARE);
- Desenvolvimento de métricas probabilísticas para detectar inconsistências (TARE);
- Adaptação do intervalo algébrico de Allen (TCI);
- Novo método para detectar inconsistência e realizar correlação temporal (TCI).

Os experimentos realizados mostraram que os métodos desenvolvidos são eficazes detectar inconsistências e popular a BC do NELL, diminuindo a quantidade de valores ausentes e

evitando a propagação de erros por meio da correção das inconsistências. Além disso, os experimentos mostraram que o componente ER reduz em mais de 30% o número de regras sem perda no processo de popular a base de conhecimento.

REFERÊNCIAS

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. In: *IN: PROCEEDINGS OF THE 1993 ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, WASHINGTON DC (USA*. [S.l.: s.n.], 1993. p. 207–216.

AGRAWAL, R.; SRIKANT, R. Mining sequential patterns. In: IEEE. *Data Engineering, 1995. Proceedings of the Eleventh International Conference on*. [S.l.], 1995. p. 3–14.

AGRAWAL, R.; SRIKANT, R. et al. Fast algorithms for mining association rules. In: *Proc. 20th int. conf. very large data bases, VLDB*. [S.l.: s.n.], 1994. v. 1215, p. 487–499.

ALE, J. M.; ROSSI, G. H. An approach to discovering temporal association rules. In: ACM. *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*. [S.l.], 2000. p. 294–300.

ALLEN, J. F. Maintaining knowledge about temporal intervals. In: . [S.l.]: ACM, 1983. v. 26, n. 11, p. 832–843.

ALON, J. et al. Discovering clusters in motion time-series data. In: IEEE. *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. [S.l.], 2003. v. 1, p. I–I.

AMARAL, B. F. d. et al. Aprimorando a classificação semissupervisionada de séries temporais extraídas de imagens de satélite. In: SOCIEDADE BRASILEIRA DE COMPUTAÇÃO-SBC. *Symposium on Knowledge Discovery, Mining and Learning, 2th*. [S.l.], 2014.

ANTUNES, C. M.; OLIVEIRA, A. L. Temporal data mining: An overview. In: *KDD workshop on temporal data mining*. [S.l.: s.n.], 2001. p. 1–13.

APPEL, A. P.; HRUSCHKA, E. Prophet–a link-predictor to learn new rules on nell. In: IEEE. *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*. [S.l.], 2011. p. 917–924.

BARALIS, E. et al. Generalized association rule mining with constraints. *Information Sciences*, Elsevier, v. 194, p. 68–84, 2012.

BATAL, I. et al. Mining recent temporal patterns for event detection in multivariate time series data. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2012. (KDD '12), p. 280–288. ISBN 978-1-4503-1462-6. Disponível em: <http://doi.acm.org/10.1145/2339530.2339578>.

BENFERHAT, S.; DUBOIS, D.; PRADE, H. Argumentative inference in uncertain and inconsistent knowledge bases. In: *Proceedings of the 9th International Conference on Uncertainty in Artificial Intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993. (UAI'93), p. 411–419. ISBN 1-55860-306-9.

BENITES, F.; SAPOZHNIKOVA, E. P. Generalized association rules for connecting biological ontologies. In: *BIOINFORMATICS*. [S.l.: s.n.], 2013. p. 229–236.

BERLINGERIO, M. et al. Mining clinical data with a temporal dimension: a case study. In: IEEE. *Bioinformatics and Biomedicine, 2007. BIBM 2007. IEEE International Conference on*. [S.l.], 2007. p. 429–436.

BERNERS-LEE, T. et al. The semantic web. *Scientific american*, New York, NY, USA:, v. 284, n. 5, p. 28–37, 2001.

BIRANT, D.; KUT, A. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, Elsevier, v. 60, n. 1, p. 208–221, 2007.

BIZER, C. et al. Dbpedia - a crystallization point for the web of data. *Web Semant.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 7, n. 3, p. 154–165, set. 2009. ISSN 1570-8268.

BOHANNON, P. et al. A cost-based model and effective heuristic for repairing constraints by value modification. In: ACM. *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. [S.l.], 2005. p. 143–154.

BOLLACKER, K. et al. Freebase: A collaboratively created graph database for structuring human knowledge. In: *Proc. of the 2008 ACM SIGMOD Int. Conf. on Management of Data*. New York, NY, USA: ACM, 2008. p. 1247–1250. ISBN 978-1-60558-102-6.

BOLLACKER, K. et al. Freebase: a collaboratively created graph database for structuring human knowledge. In: ACM. *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. [S.l.], 2008. p. 1247–1250.

BRISSON, L.; COLLARD, M.; PASQUIER, N. Improving the knowledge discovery process using ontologies. In: *IEEE MCD'2005 international workshop on Mining Complex Data*. [S.l.: s.n.], 2005. p. 25–32.

BURDICK, D.; CALIMLIM, M.; GEHRKE, J. Mafia: A maximal frequent itemset algorithm for transactional databases. In: IEEE. *Data Engineering, 2001. Proceedings. 17th International Conference on*. [S.l.], 2001. p. 443–452.

BURTON, S. H. et al. Mining useful association rules from questionnaire data. *Intell. Data Anal.*, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 18, n. 3, p. 479–494, maio 2014. ISSN 1088-467X. Disponível em: <http://dx.doi.org/10.3233/IDA-140652>.

CALDERS, T.; GOETHALS, B.; MAMPAEY, M. Mining itemsets in the presence of missing values. In: ACM. *Proceedings of the 2007 ACM symposium on Applied computing*. [S.l.], 2007. p. 404–408.

CARIÑENA, P. Fuzzy temporal association rules: combining temporal and quantitative data to increase rule expressiveness. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, Wiley Online Library, v. 4, n. 1, p. 64–70, 2014.

CARLSON, A. et al. Coupling semi-supervised learning of categories and relations. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the NAACL HLT 2009 Workshop on Semi-supervised Learning for Natural Language Processing*. [S.l.], 2009. p. 1–9.

CARLSON, A. et al. Toward an architecture for never-ending language learning. In: *In AAAI*. [S.l.: s.n.], 2010.

CARLSON, A. et al. Coupled semi-supervised learning for information extraction. In: ACM. *Proceedings of the third ACM international conference on Web search and data mining*. [S.l.], 2010. p. 101–110.

CHAN, K.-P.; FU, A. W.-C. Efficient time series matching by wavelets. In: IEEE. *Data Engineering, 1999. Proceedings., 15th International Conference on*. [S.l.], 1999. p. 126–133.

CHEN, R.-H.; FAN, C.-M. Treatment of missing values for association rule-based tool commonality analysis in semiconductor manufacturing. In: IEEE. *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*. [S.l.], 2012. p. 886–891.

CHEN, X.-y.; YE, D.-y.; HU, X.-L. Entropy-based symbolic representation for time series classification. In: IEEE. *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*. [S.l.], 2007. v. 2, p. 754–760.

CHIANG, F.; MILLER, R. J. A unified model for data and constraint repair. In: IEEE. *Data Engineering (ICDE), 2011 IEEE 27th International Conference on*. [S.l.], 2011. p. 446–457.

CHU, X. et al. Katara: A data cleaning system powered by knowledge bases and crowdsourcing. In: ACM. *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. [S.l.], 2015. p. 1247–1261.

CRUZ-FILIPE, L. et al. repairc: A tool for ensuring data consistency by means of active integrity constraints. *arXiv preprint arXiv:1510.03989*, 2015.

DAS, G.; GUNOPULOS, D.; MANNILA, H. Finding similar time series. In: SPRINGER. *European Symposium on Principles of Data Mining and Knowledge Discovery*. [S.l.], 1997. p. 88–100.

DAS, G. et al. Rule discovery from time series. In: *KDD*. [S.l.: s.n.], 1998. v. 98, p. 16–22.

DJENOURI, Y.; DRIAS, H.; BENDJOUDI, A. Pruning irrelevant association rules using knowledge mining. *International Journal of Business Intelligence and Data Mining*, Inderscience Publishers Ltd, v. 9, n. 2, p. 112–144, 2014.

DOMINGOS, P.; LOWD, D. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, Morgan & Claypool Publishers, v. 3, n. 1, p. 1–155, 2009.

DONG, X. et al. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: ACM. *Proc. of the 20th ACM SIGKDD int. conf. on Knowledge discovery and data mining*. [S.l.], 2014. p. 601–610.

DONG, X. et al. Knowledge vault: A web-scale approach to probabilistic knowledge fusion. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2014. (KDD '14), p. 601–610. ISBN 978-1-4503-2956-9. Disponível em: <http://doi.acm.org/10.1145/2623330.2623623>.

DYLLA, M.; MILIARAKI, I.; THEOBALD, M. A temporal-probabilistic database model for information extraction. *Proc. of the VLDB Endowment*, VLDB Endowment, v. 6, n. 14, p. 1810–1821, 2013.

DYLLA, M.; MILIARAKI, I.; THEOBALD, M. A temporal-probabilistic database model for information extraction. *Proc. VLDB Endow.*, VLDB Endowment, v. 6, n. 14, p. 1810–1821, set. 2013. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2556549.2556564>.

ESCOVAR, E. L.; YAGUINUMA, C. A.; BIAJIZ, M. Using fuzzy ontologies to extend semantically similar data mining. In: *SBBD*. [S.l.: s.n.], 2006. p. 16–30.

FALOUTSOS, C.; RANGANATHAN, M.; MANOLOPOULOS, Y. *Fast subsequence matching in time-series databases*. [S.l.]: ACM, 1994.

FAN, M.; ZHOU, Q.; ZHENG, T. F. Learning embedding representations for knowledge inference on imperfect and incomplete repositories. *arXiv preprint arXiv:1503.08155*, 2015.

FAN, W. et al. Towards certain fixes with editing rules and master data. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 3, n. 1-2, p. 173–184, 2010.

FIKES, R.; HAYES, P.; HORROCKS, I. Owl-ql?a language for deductive query answering on the semantic web. *Web semantics: Science, services and agents on the World Wide Web*, Elsevier, v. 2, n. 1, p. 19–29, 2004.

FISHER, D. H. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, Springer, v. 2, n. 2, p. 139–172, 1987.

FLEISCHMAN, M.; DECAMP, P.; ROY, D. Mining temporal patterns of movement for video content classification. In: *Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval*. New York, NY, USA: ACM, 2006. (MIR '06), p. 183–192. ISBN 1-59593-495-2. Disponível em: <http://doi.acm.org/10.1145/1178677.1178704>.

FOURNIER-VIGER, P. et al. Spmf: a java open-source pattern mining library. *Journal of Machine Learning Research*, v. 15, n. 1, p. 3389–3393, 2014.

FOURNIER-VIGER, P.; TSENG, V. S. Mining top-k non-redundant association rules. In: *Foundations of Intelligent Systems*. [S.l.]: Springer, 2012. p. 31–40.

FRADKIN, D.; MÖRCHEN, F. Mining sequential patterns for classification. *Knowledge and Information Systems*, Springer, p. 1–19, 2015.

FUJITA, A. et al. Functional clustering of time series gene expression data by granger causality. *BMC systems biology*, BioMed Central, v. 6, n. 1, p. 137, 2012.

GALÁRRAGA, L. A. et al. Amie: Association rule mining under incomplete evidence in ontological knowledge bases. In: *Proceedings of the 22Nd Int. Conf. on World Wide Web*. Republic and Canton of Geneva, Switzerland: Int. World Wide Web Conf. Steering Committee, 2013. (WWW '13), p. 413–422. ISBN 978-1-4503-2035-1. Disponível em: <http://dl.acm.org/citation.cfm?id=2488388.2488425>.

GARDNER, M. et al. Improving learning and inference in a large knowledge-base using latent syntactic cues. In: *EMNLP*. [S.l.: s.n.], 2013. p. 833–838.

GOUDA, K.; ZAKI, M. J. Genmax: An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 11, n. 3, p. 223–242, nov. 2005. ISSN 1384-5810. Disponível em: <http://dx.doi.org/10.1007/s10618-005-0002-x>.

GRAHNE, G.; ZHU, J. High performance mining of maximal frequent itemsets. In: *6th International Workshop on High Performance Data Mining*. [S.l.: s.n.], 2003.

GRANGER, C. W. J. Investigating causal relationships by econometric models and cross-spectral methods', economˆ trica. *Econometrica*, [Wiley, Econometric Society], v. 37, n. 3, p. 424–438, 1969.

GRAU, B. C. et al. Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, Elsevier, v. 6, n. 4, p. 309–322, 2008.

GRUBER, T. R. A translation approach to portable ontology specifications. *Knowledge acquisition*, Elsevier, v. 5, n. 2, p. 199–220, 1993.

GRZYMALA-BUSSE, J. W.; HU, M. A comparison of several approaches to missing attribute values in data mining. In: SPRINGER. *Rough sets and current trends in computing*. [S.l.], 2001. p. 378–385.

GUARINO, N. *Formal ontology in information systems: Proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*. [S.l.]: IOS press, 1998.

GUARINO, N.; GIARETTA, P.; MARS, N. Towards very large knowledge bases: Knowledge building and knowledge sharing. ontologies and knowledge bases: Towards a terminological clarification. n. *Mars. Amsterdam, IOS Press*, p. 25–32, 1995.

HAN, J.; KAMBER, M.; PEI, J. *Data mining, southeast asia edition: Concepts and techniques*. [S.l.]: Morgan kaufmann, 2006.

HAN, J.; PEI, J.; YIN, Y. Mining frequent patterns without candidate generation. In: ACM. *ACM sigmod record*. [S.l.], 2000. v. 29, n. 2, p. 1–12.

HASHIKAMI, H.; KODA, M. Proposal of new objective measures for mining association rules: Cannibalization and unexpectedness. *Annals of Data Science*, Springer, v. 1, n. 1, p. 25–39, 2014.

HAUTALA, J.; JAUHIAINEN, J. S. Spatio-temporal processes of knowledge creation. *Research Policy*, Elsevier, v. 43, n. 4, p. 655–668, 2014.

HELLMANN, S. et al. Knowledge base creation, enrichment and repair. In: *Linked Open Data–Creating Knowledge Out of Interlinked Data*. [S.l.]: Springer, 2014. p. 45–69.

HOFFART, J. et al. Yago2: A spatially and temporally enhanced knowledge base from wikipedia (extended abstract). In: *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*. AAAI Press, 2013. (IJCAI '13), p. 3161–3165. ISBN 978-1-57735-633-2. Disponível em: <http://dl.acm.org/citation.cfm?id=2540128.2540600>.

HONG, T.-P.; WU, C.-W. Mining rules from an incomplete dataset with a high missing rate. *Expert Systems with Applications*, Elsevier, v. 38, n. 4, p. 3931–3936, 2011.

JI, H. et al. Tackling representation, annotation and classification challenges for temporal knowledge base population. *Knowledge and Information Systems*, Springer, v. 41, n. 3, p. 611–646, 2014.

JR, R. J. B. Efficiently mining long patterns from databases. In: ACM. *ACM Sigmod Record*. [S.l.], 1998. v. 27, n. 2, p. 85–93.

KEOGH, E. J.; PAZZANI, M. J. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In: *KDD*. [S.l.: s.n.], 1998. v. 98, p. 239–243.

KEOGH, E. J.; SMYTH, P. A probabilistic approach to fast pattern matching in time series databases. In: *KDD*. [S.l.: s.n.], 1997. v. 1997, p. 24–30.

KETTERLIN, A. Clustering sequences of complex objects. In: *KDD*. [S.l.: s.n.], 1997. p. 215–218.

KIM, J. W. Construction and evaluation of structured association map for visual exploration of association rules. *Expert Systems with Applications*, Elsevier, v. 74, p. 70–81, 2017.

KNUBLAUCH, H. et al. The protégé owl plugin: An open development environment for semantic web applications. In: *The Semantic Web–ISWC 2004*. [S.l.]: Springer, 2004. p. 229–243.

KOUBARAKIS, M. Reasoning about time and change: A knowledge base management perspective. *University of Toronto, Canada*, Citeseer, 1990.

KRYSZKIEWICZ, M. Association rules in incomplete databases. In: SPRINGER. *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. [S.l.], 1999. p. 84–93.

KUZEY, E.; WEIKUM, G. Extraction of temporal facts and events from wikipedia. In: *Proceedings of the 2Nd Temporal Web Analytics Workshop*. New York, NY, USA: ACM, 2012. (TempWeb '12), p. 25–32. ISBN 978-1-4503-1188-5. Disponível em: <http://doi.acm.org/10.1145/2169095.2169101>.

LACASTA, J. et al. Population of a spatio-temporal knowledge base for jurisdictional domains. *International Journal of Geographical Information Science*, Taylor & Francis, v. 28, n. 9, p. 1964–1987, 2014.

LAO, N.; COHEN, W. W. Relational retrieval using a combination of path-constrained random walks. *Machine learning*, Springer, v. 81, n. 1, p. 53–67, 2010.

LAXMAN, S.; SASTRY, P. S. A survey of temporal data mining. *Sadhana*, Springer India, in co-publication with Indian Academy of Sciences, v. 31, n. 2, p. 173–198, 2006.

LEE, C.-H. Mining spatio-temporal information on microblogging streams using a density-based online clustering method. *Expert Systems with Applications*, Elsevier, v. 39, n. 10, p. 9623–9641, 2012.

LEHMANN, J.; BÜHMANN, L. Ore-a tool for repairing and enriching knowledge bases. *The Semantic Web–ISWC 2010*, Springer, p. 177–193, 2010.

LESH, N.; ZAKI, M. J.; OGIHARA, M. Mining features for sequence classification. In: ACM. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 1999. p. 342–346.

LI, H. et al. Event specific multimodal pattern mining for knowledge base construction. In: ACM. *Proceedings of the 2016 ACM on Multimedia Conference*. [S.l.], 2016. p. 821–830.

LIAN, X. et al. Mining spatial and spatio-temporal rois for action recognition. 2016.

LIBEN-NOWELL, D.; KLEINBERG, J. The link-prediction problem for social networks. *journal of the Association for Information Science and Technology*, Wiley Online Library, v. 58, n. 7, p. 1019–1031, 2007.

MA, Y. et al. Learning disjointness axioms with association rule mining and its application to inconsistency detection of linked data. In: SPRINGER. *Chinese Semantic Web and Web Science Conference*. [S.l.], 2014. p. 29–41.

MAHDISOLTANI, F.; BIEGA, J.; SUCHANEK, F. Yago3: A knowledge base from multilingual wikipedias. In: CIDR 2015. *7th Biennial Conference on Innovative Data Systems Research*. [S.l.], 2014.

MANNILA, H.; MEEK, C. Global partial orders from sequential data. In: ACM. *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 2000. p. 161–168.

MANNILA, H.; TOIVONEN, H. Discovering generalized episodes using minimal occurrences. In: *KDD*. [S.l.: s.n.], 1996. v. 96, p. 146–151.

MARINICA, C.; GUILLET, F. Knowledge-based interactive postmining of association rules using ontologies. *IEEE Transactions on Knowledge and Data Engineering*, IEEE, v. 22, n. 6, p. 784–797, 2010.

MATUSZEK, C. et al. An introduction to the syntax and content of cyc. In: *Proceedings of the 2006 AAAI Spring Symposium on Formalizing and Compiling Background Knowledge and Its Applications to Knowledge Representation and Question Answering*. [S.l.: s.n.], 2006. p. 44–49.

MAYFIELD, C.; NEVILLE, J.; PRABHAKAR, S. Eracer: a database approach for statistical inference and data cleaning. In: ACM. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. [S.l.], 2010. p. 75–86.

MCGUINNESS, D. L.; HARMELEN, F. V. et al. Owl web ontology language overview. *W3C recommendation*, v. 10, n. 10, p. 2004, 2004.

MENDES, P. N.; JAKOB, M.; BIZER, C. Dbpedia: A multilingual cross-domain knowledge base. In: *LREC*. [S.l.: s.n.], 2012. p. 1813–1817.

MIANI, R. G. et al. Narfo algorithm: Mining non-redundant and generalized association rules based on fuzzy ontologies. In: *Enterprise Inf. Systems*. [S.l.]: Springer, 2009. p. 415–426.

MIANI, R. G. et al. Narfo* algorithm-optimizing the process of obtaining non-redundant and generalized semantic association rules. In: *ICEIS (2)*. [S.l.: s.n.], 2010. p. 320–325.

MIANI, R. G. L.; HRUSCHKA JUNIOR, E. R. Exploring association rules in a large growing knowledge base. *Int. J. of Comp. Info. Syst. and Ind. Mangt Apps*, p. 106–114, 2015.

MIANI, R. G. L.; HRUSCHKA JUNIOR, E. R. Correcting inconsistencies trough association rules in temporal large knowledge bases. In: *6th BRACIS*. [S.l.]: IEEE, 2017. p. 1–6.

MIANI, R. G. L.; HRUSCHKA JUNIOR, E. R. Specific temporal association rules and temporal correlations to enlarge and detect inconsistencies in a large growing knowledge base. In: *13th ICNC-FSKD*. [S.l.]: IEEE, 2017. p. 1537–1546.

MIANI, R. G. L.; PEDRO, S. D. d. S.; HRUSCHKA JUNIOR, E. R. Association rules to help populating a never-ending growing knowledge base. In: *Advances in Artificial Intelligence–IBERAMIA 2014*. [S.l.]: Springer, 2014. p. 169–181.

MITSA, T. *Temporal data mining*. [S.l.]: CRC Press, 2010.

MOHAMED, T. P.; HRUSCHKA JUNIOR, E. R.; MITCHELL, T. M. Discovering relations between noun categories. In: ASSOC. FOR COMP. LINGUISTICS. *Proc. of the Conf. on Empirical Methods in Natural Language Processing*. [S.l.], 2011. p. 1447–1455.

MOSKOVITCH, R.; SHAHAR, Y. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowledge and Information Systems*, Springer, p. 1–40, 2014.

NAKASHOLE, N.; THEOBALD, M.; WEIKUM, G. Scalable knowledge harvesting with high precision and high recall. In: ACM. *Proceedings of the fourth ACM int. conf. on Web search and data mining*. [S.l.], 2011. p. 227–236.

NARDI, D.; BRACHMAN, R. J. et al. An introduction to description logics. In: *Description logic handbook*. [S.l.: s.n.], 2003. p. 1–40.

NAYAK, J. R.; COOK, D. J. Approximate association rule mining. In: *FLAIRS Conference*. [S.l.: s.n.], 2001. p. 259–263.

NEELAKANTAN, A.; CHANG, M.-W. Inferring missing entity type instances for knowledge base completion: New dataset and methods. *arXiv preprint arXiv:1504.06658*, 2015.

NGOMO, A.-C. N.; SHERIF, M. A.; LYKO, K. Unsupervised link discovery through knowledge base repair. In: SPRINGER. *European Semantic Web Conference*. [S.l.], 2014. p. 380–394.

NICOLETTI, M. C. et al. Representation and automatic learning of temporal relations between time periods with uncertain boundaries. In: IEEE. *Intelligent Systems Design and Applications (ISDA), 2012 12th International Conference on*. [S.l.], 2012. p. 443–448.

OATES, T. Identifying distinctive subsequences in multivariate time series by clustering. In: ACM. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. [S.l.], 1999. p. 322–326.

OZDEN, B.; RAMASWAMY, S.; SILBERSCHATZ, A. Cyclic association rules. In: IEEE. *Data Engineering, 1998. Proceedings., 14th International Conference on*. [S.l.], 1998. p. 412–421.

PARK, J. S.; CHEN, M.-S.; YU, P. S. *An effective hash-based algorithm for mining association rules*. [S.l.]: ACM, 1995.

PASQUIER, N. et al. Discovering frequent closed itemsets for association rules. In: *Proceedings of the 7th International Conference on Database Theory*. London, UK, UK: Springer-Verlag, 1999. (ICDT '99), p. 398–416. ISBN 3-540-65452-6. Disponível em: <http://dl.acm.org/citation.cfm?id=645503.656256>.

PATEL, D.; HSU, W.; LEE, M. L. Mining relationships among interval-based events for classification. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. New York, NY, USA: ACM, 2008. (SIGMOD '08), p. 393–404. ISBN 978-1-60558-102-6. Disponível em: <http://doi.acm.org/10.1145/1376616.1376658>.

PAULHEIM, H. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic web*, IOS Press, v. 8, n. 3, p. 489–508, 2017.

PAULHEIM, H.; BIZER, C. Improving the quality of linked data using statistical distributions. *International Journal on Semantic Web and Information Systems (IJSWIS)*, IGI Global, v. 10, n. 2, p. 63–86, 2014.

PEDRO, S. D.; HRUSCHKA JUNIOR, E. R. Collective intelligence as a source for machine learning self-supervision. In: ACM. *Proceedings of the 4th International Workshop on Web Intelligence & Communities*. [S.l.], 2012. p. 5.

PEDRO, S. D.; HRUSCHKA JUNIOR, E. R. Conversing learning: Active learning and active social interaction for human supervision in never-ending learning systems. In: *Advances in Artificial Intelligence–IBERAMIA 2012*. [S.l.]: Springer, 2012. p. 231–240.

PEREGO, R.; ORLANDO, S.; PALMERINI, P. Enhancing the apriori algorithm for frequent set counting. In: *Data Warehousing and Knowledge Discovery*. [S.l.]: Springer, 2001. p. 71–82.

QAMRA, A.; TSENG, B.; CHANG, E. Y. Mining blog stories using community-based and temporal clustering. In: ACM. *Proceedings of the 15th ACM international conference on Information and knowledge management*. [S.l.], 2006. p. 58–67.

QUINLAN, J.; CAMERON-JONES, R. Foil: A midterm report. In: SPRINGER. *Machine Learning: ECML-93*. [S.l.], 1993. p. 1–20.

RAGEL, A. et al. Treatment of missing values for association rules. In: *Research and Development in Knowledge Discovery and Data Mining*. [S.l.]: Springer, 1998. p. 258–270.

RAMAN, V.; HELLERSTEIN, J. M. Potter's wheel: An interactive data cleaning system. In: *VLDB*. [S.l.: s.n.], 2001. v. 1, p. 381–390.

RAMASWAMY, S.; MAHAJAN, S.; SILBERSCHATZ, A. On the discovery of interesting patterns in association rules. In: CITESEER. *VLDB*. [S.l.], 1998. v. 98, p. 368–379.

RAMESHKUMAR, K.; SAMBATH, M.; RAVI, S. Relevant association rule mining from medical dataset using new irrelevant rule elimination technique. In: IEEE. *Information Communication and Embedded Systems (ICICES), 2013 International Conference on*. [S.l.], 2013. p. 300–304.

SARASUA, C.; SIMPERL, E.; NOY, N. F. Crowdmap: Crowdsourcing ontology alignment with microtasks. In: *The Semantic Web–ISWC 2012*. [S.l.]: Springer, 2012. p. 525–541.

SEFIDMAZGI, M. G. et al. Trend analysis using non-stationary time series clustering based on the finite element method. *Nonlinear Processes in Geophysics*, Copernicus GmbH, v. 21, n. 3, p. 605–615, 2014.

SHAHEEN, M.; SHAHBAZ, M.; GUERGACHI, A. Context based positive and negative spatio-temporal association rule mining. *Knowledge-based systems*, Elsevier, v. 37, p. 261–273, 2013.

SHIN, J. et al. Incremental knowledge base construction using deepdive. *Proceedings of the VLDB Endowment*, VLDB Endowment, v. 8, n. 11, p. 1310–1321, 2015.

SIEGEL, N. et al. The cyc® system: Notes on architecture. *Cycorp, Inc. Retrieved October*, v. 10, p. 9, 2004.

SMYTH, P. Probabilistic model-based clustering of multivariate and sequential data. In: SAN FRANCISCO, CA: MORGAN KAUFMAN. *Proceedings of the Seventh International Workshop on AI and Statistics*. [S.l.], 1999. p. 299–304.

SMYTH, P. et al. Clustering sequences with hidden markov models. *Advances in neural information processing systems*, Citeseer, p. 648–654, 1997.

SRIKANT, R.; AGRAWAL, R. Mining generalized association rules. In: *Proceedings of the 21th Int. Conf. on Very Large Data Bases*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995. (VLDB '95), p. 407–419. ISBN 1-55860-379-4. Disponível em: <http://dl.acm.org/citation.cfm?id=645921.673304>.

SRIKANT, R.; AGRAWAL, R. Mining sequential patterns: Generalizations and performance improvements. *Advances in Database Technology?EDBT'96*, Springer, p. 1–17, 1996.

SUCHANEK, F. M.; KASNECI, G.; WEIKUM, G. Yago: A core of semantic knowledge. In: *Proceedings of the 16th International Conference on World Wide Web*. New York, NY, USA: ACM, 2007. (WWW '07), p. 697–706. ISBN 978-1-59593-654-7. Disponível em: <http://doi.acm.org/10.1145/1242572.1242667>.

SZVARÇA, R. R. et al. Regras de associação temporal em cancer de mama. *Revista de Engenharia e Tecnologia*, v. 7, n. 4, p. Páginas–84, 2016.

TALHA, A. M.; JUNEJO, I. N. Dynamic scene understanding using temporal association rules. *Image and Vision Computing*, Elsevier, v. 32, n. 12, p. 1102–1116, 2014.

TALUKDAR, P. P.; WIJAYA, D.; MITCHELL, T. Coupled temporal scoping of relational facts. In: ACM. *Proceedings of the fifth ACM international conference on Web search and data mining*. [S.l.], 2012. p. 73–82.

TANON, T. P. et al. From freebase to wikidata: The great migration. In: INTERNATIONAL WORLD WIDE WEB CONFERENCES STEERING COMMITTEE. *Proceedings of the 25th International Conference on World Wide Web*. [S.l.], 2016. p. 1419–1428.

TSENG, V. S.; LEE, C.-H. Effective temporal data classification by integrating sequential pattern mining and probabilistic induction. *Expert Systems with Applications*, Elsevier, v. 36, n. 5, p. 9524–9532, 2009.

VANNELLA, D. et al. Validating and extending semantic knowledge bases using video games with a purpose. In: *Proc. of ACL*. [S.l.: s.n.], 2014.

VO, B.; HONG, T.-P.; LE, B. A lattice-based approach for mining most generalization association rules. *Know.-Based Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 45, p. 20–30, jun. 2013. ISSN 0950-7051. Disponível em: <http://dx.doi.org/10.1016/j.knosys.2013.02.003>.

VO, B.; LE, B. Mining the most generalization association rules. In: *Advances in Intelligent Information and Database Systems*. [S.l.]: Springer, 2010. p. 207–216.

VRANDEČIĆ, D.; KRÖTZSCH, M. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, ACM, v. 57, n. 10, p. 78–85, 2014.

WANG, R. C.; COHEN, W. W. Character-level analysis of semi-structured documents for set expansion. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*. [S.l.], 2009. p. 1503–1512.

WANG, Y. et al. Harvesting facts from textual web sources by constrained label propagation. In: ACM. *Proc. of the 20th ACM int. conf. on Information and knowledge management*. [S.l.], 2011. p. 837–846.

WANG, Y. et al. Scalable spatio-temporal knowledge harvesting. In: ACM. *Proceedings of the 20th international conference companion on World wide web*. [S.l.], 2011. p. 143–144.

WANG, Y. et al. Timely yago: harvesting, querying, and visualizing temporal knowledge from wikipedia. In: ACM. *Proceedings of the 13th International Conference on Extending Database Technology*. [S.l.], 2010. p. 697–700.

WEBB, G. I.; ZHANG, S. Removing trivial associations in association rule discovery. In: *Abstracts Published in the Proceedings of the First International NAISO Congress on Autonomous Intelligent Systems (ICAIS 2002)*. [S.l.: s.n.], 2002.

WINARKO, E.; RODDICK, J. F. Armada–an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, Elsevier, v. 63, n. 1, p. 76–90, 2007.

WOJCIECHOWSKI, M.; ZAKRZEWICZ, M. On efficiency of dataset filtering implementations in constraint-based discovery of frequent itemsets. In: CITESEER. *JCKBSE Conference. Maribor, Slovenia*. [S.l.], 2002.

YAKOUT, M.; BERTI-ÉQUILLE, L.; ELMAGARMID, A. K. Don't be scared: use scalable automatic repairing with maximal likelihood and bounded changes. In: ACM. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. [S.l.], 2013. p. 553–564.

YANG, B.; HUANG, H. Topsil-miner: an efficient algorithm for mining top-k significant itemsets over data streams. *Knowledge and information systems*, Springer, v. 23, n. 2, p. 225–242, 2010.

YEN, S.-J.; CHEN, A. L. An efficient approach to discovering knowledge from large databases. In: IEEE. *Parallel and Distributed Information Systems, 1996., Fourth International Conference on*. [S.l.], 1996. p. 8–18.

ZAKI, M. J. Generating non-redundant association rules. In: *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2000. (KDD '00), p. 34–43. ISBN 1-58113-233-6. Disponível em: <http://doi.acm.org/10.1145/347090.347101>.

ZAKI, M. J.; HSIAO, C.-J. Charm: An efficient algorithm for closed itemset mining. In: SIAM. *Proceedings of the 2002 SIAM international conference on data mining*. [S.l.], 2002. p. 457–473.

ZHANG, Y.; EICK, C. F. Novel clustering and analysis techniques for mining spatio-temporal data. In: ACM. *Proceedings of the 1st ACM SIGSPATIAL PhD Workshop*. [S.l.], 2014. p. 2.

GLOSSÁRIO

ARs – *Antecedent Rules*

BCs – *Bases de Conhecimento*

CL – *Conversing Learning*

CMC – *Coupled Morphological Classifier*

CPL – *Coupled Pattern Learner*

CRs – *Consequent Rules*

CSEAL – *Coupled SEAL*

DL – *Description Logics*

ER – *Eliminating Rules*

FCI – *Frequent Closed Itemset*

FI – *Frequent Itemset*

GBC – *Grandes Bases de Conhecimento*

HMM – *Hidden Markov Model*

MFI – *Maximal Frequent Itemset*

MGAR – *Most Generalization Association Rules*

MP1 – *Métrica Probabilística 1*

MP1 – *Métrica Probabilística 2*

MSC – *Modified Support Calculation*

NELL – *Never-Ending Language Learning*

OWL – *Web Ontology Language*

OntExt – *Ontology Extension System*

PRA – *Path Ranking Algorithm*

RAG – *Regras de Associacão Generalizadas*

RA – *Regras de Associacão*

SEAL – *Set Expander for Any Language*

STARs – *Specific Temporal Association Rules*

TARE – *Temporal Association Rules Extraction*

TIA – *Temporal Interval Algebra*

URI – *Uniform Resource Identifier*