JOÃO ANTONIO DA SILVA

UMA ABORDAGEM INCREMENTAL PARA RESOLUÇÃO DE ENTIDADES DESCRITAS POR DADOS TEXTUAIS CURTOS

LAVRAS – MG

2017

JOÃO ANTONIO DA SILVA

UMA ABORDAGEM INCREMENTAL PARA RESOLUÇÃO DE ENTIDADES DESCRITAS POR DADOS TEXTUAIS CURTOS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

Prof. Denilson Alves Pereira
Orientador

LAVRAS – MG
2017

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo(a) próprio(a) autor(a).

Silva, João Antonio.
Uma Abordagem Incremental para Resolução de Entidades Descritas por Dados Textuais Curtos / João Antonio da Silva. 2017.
116 p.

Orientador: Prof. Denilson Alves Pereira.

Dissertação (mestrado acadêmico) - Universidade Federal de Lavras, 2017.
Bibliografia.

1. Resolução de Entidades. 2. Classificação Associativa. 3. Aprendizagem de Máquina. I. Pereira, Denilson Alves. II. Título.

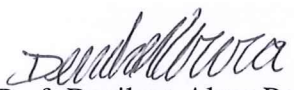
JOÃO ANTONIO DA SILVA

UMA ABORDAGEM INCREMENTAL PARA RESOLUÇÃO DE ENTIDADES DESCRITAS POR DADOS TEXTUAIS CURTOS

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação para a obtenção do título de Mestre em Ciência da Computação.

APROVADA em 22 de Fevereiro de 2017.

Prof. Denilson Alves Pereira	DCC - UFLA
Prof. Álvaro Rodrigues Pereira Júnior	DECOM - UFOP
Prof. Thierson Couto Rosa	Instituto de Informática - UFG



Prof. Denilson Alves Pereira
Orientador

LAVRAS – MG
2017

Dedico esta conquista à minha mãe Eliana, e a meus irmãos Ana Paula e Pedro.
Por toda força recebida e por acreditarem que mesmo com todas as dificuldades,
eu seria capaz.

AGRADECIMENTOS

Em primeiro lugar a Deus, por minha vida e por estar ao meu lado me iluminando. Obrigado.

À minha mãe Eliana, por todo amor incondicional. Mesmo com a distância, sempre presente, obrigado por me manter focado, e me fazer acreditar que sou capaz, mesmo quando eu próprio não acredito.

Obrigado a meus irmãos Ana Paula e Pedro, por toda a compreensão e suporte.

Agradeço à Universidade Federal de Lavras, ao Departamento de Ciência da Computação e ao Programa de Pós-Graduação em Ciência da Computação da UFLA, por todos os recursos disponibilizados para a realização deste trabalho. Obrigado pela oportunidade.

Agradeço à CAPES pelo apoio financeiro, fundamental para a realização e conclusão do Mestrado, e à FAPEMIG, por meio do projeto CEX-APQ-01834-14.

Ao Prof. Denilson, pela orientação durante a realização deste trabalho. Obrigado.

Obrigado aos professores Álvaro R. P. Júnior (DECOM - UFOP) e Thierson C. Rosa (Instituto de Informática - UFG) pela disponibilidade para participação e avaliação deste trabalho. Obrigado por todas as contribuições.

Agradeço à toda equipe do LabRI, por todas as discussões e por todo o conhecimento compartilhado.

Obrigado a todos professores, colegas, secretárias e funcionários do Departamento de Ciência da Computação da UFLA, por todo conhecimento, suporte e apoio.

A meus amigos e colegas, por toda a disponibilidade sempre que precisei.

A todos que contribuíram, torceram e acreditaram em minha conquista, direta ou indiretamente.

Obrigado. Muito obrigado a todos!

"Use me, God.

Show me how to take who I am, who I want to be, and what I can do,

and use it for a purpose greater than myself."

Martin Luther King Jr.

RESUMO

Diversas aplicações Web mantêm repositórios de dados com referências a milhares de entidades do mundo real. Esses dados têm origem em diversas fontes e novos dados continuamente são agregados a esses repositórios. Identificar o conjunto de entidades distintas e associar as referências corretamente a cada entidade é um problema conhecido como resolução de entidades. Atualmente, um desafio é resolver esse problema incrementalmente, à medida que novos dados se tornam disponíveis, especialmente quando os registros de dados são descritos por um único atributo textual. Neste trabalho, é proposta uma abordagem incremental para resolução de entidades. Diferente de abordagens tradicionais o método implementado, denominado AssocIER, usa um *ensemble* de classificadores multiclasses com auto treinamento e detecção de novas classes para incrementalmente agrupar referências à mesma entidade. O auto treinamento da abordagem permite a atualização automática do modelo de aprendizagem na fase de predição, enquanto o mecanismo de detecção de novas classes permite a identificação de registros de classes desconhecidas em tempo de treinamento. O principal classificador no *ensemble* é um caso particular de classificador associativo, que pode ser implementado eficientemente. A abordagem proposta foi avaliada em várias bases de dados reais e diferentes cenários, e foi comparada com uma abordagem tradicional para a resolução de entidades. Os resultados obtidos mostram que o AssocIER é efetivo e eficiente na solução de entidades cujos dados são não estruturados e na presença de um número muito alto de entidades reais distintas, sendo capaz de identificar centenas de novas classes. Os resultados também mostram que o AssocIER pode melhorar muito a performance em base de dados cujos registros são ofertas de produtos, tipo de dados que o *baseline* não apresenta bons resultados. Nesse caso, os resultados obtidos chegam a ser 149% mais efetivos e chega a ser 385 vezes mais rápido na fase de predição. Os resultados ainda demostram a importância da incorporação de novos dados no modelo de aprendizagem, principalmente quando a base de dados contém poucos registros por classe. Ademais, a abordagem proposta apresenta bom comportamento quando poucos registros estão disponíveis para a geração de uma solução inicial, sendo mesmo possível sua execução sem nenhum dado de treinamento, caso em que o modelo de aprendizagem é totalmente gerado incrementalmente na fase de teste.

Palavras-chave: Resolução de Entidades, Classificação Associativa, Aprendizagem Incremental.

ABSTRACT

Several Web applications maintain data repositories containing references to thousands of real-world entities originating from multiple sources, and they continually receive new data. Identifying the distinct entities and associating the correct references to each one is a problem known as entity resolution. The challenge is to solve the problem incrementally, as the data arrive, especially when those data are described by a single textual attribute. In this work, we propose a approach for incremental entity resolution. Unlike traditional approaches, the method we implemented, called AssocIER, uses an ensemble of multiclass classifiers with self-training and detection of novel classes to incrementally group entity references. Self-training allows the learning model to be automatically updated during the prediction phase, and the novel class detection mechanism allows the identification of records of unknown classes in the training time. Our main classifier is based on a restricted case of association rules, which can be implemented efficiently. We evaluated our method in various real-world datasets and scenarios, comparing it with a traditional entity resolution approach. The results show that AssocIER is effective and efficient to solve unstructured data in collections with a very large number of entities and features, and is able to detect hundreds of novel classes. We found that AssocIER can greatly improve the performance of resolving product data, which is a weakness of the baseline, achieving gains of 149% in effectiveness and being up to 385 times faster in the prediction phase. The results also show that it is important to incorporate new data into the learning model, especially for datasets with fewer records per class. Furthermore, our method behaves well in scenarios of scarce availability of examples for training, being able to run even with no training data.

Keywords: Entity Resolution, Associative Classification, Incremental Learning.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ACRÔNIMOS

AssocIER: *Associative Classifier for Incremental Entity Resolution*

ER: *Entity Resolution*

FMC: Fragmentação Média da Classe

KNN: *k-Nearest Neighbor*

PMC: Pureza Média da Classe

SVM: *Suport Vector Machine*

TER: *Traditional Entity Resolution*

TF-IDF: *Term Frequency* and *Inverse Document Frequency*

SUMÁRIO

1 INTRODUÇÃO

Dados dois conjuntos de registros $A \subseteq I_1$ e $B \subseteq I_2$, de duas bases de dados I_1 e I_2, a resolução de entidades - *Entity Resolution* (ER) - consiste em identificar todas as referências de $A \times B$ que correspondem à mesma entidade do mundo real (GETOOR; MACHANAVAJJHALA, 2012; KÖPCKE; RAHM, 2010; BENJELLOUN et al., 2009; ELMAGARMID; IPEIROTIS; VERYKIOS, 2007; KOUDAS; SARAWAGI; SRIVASTAVA, 2006). Essa definição inclui o caso particular em que $I_1 = I_2$, e assim a busca por referências que correspondem à mesma entidade ocorre na mesma base de dados. A resolução de entidades é um importante passo na integração de bases de dados para assegurar a qualidade dos dados.

O problema de resolução de entidades pode ser observado em várias aplicações. Um bom exemplo são as listas de correspondência, em que as informações dos destinatários podem estar armazenadas como várias entradas na base de dados, cada uma com pequenas diferenças. Em bibliotecas digitais, o problema de resolução de entidades pode ser observado na resolução de referências bibliográficas, isto é, na identificação de quais referências correspondem à mesma publicação (CARVALHO et al., 2006). Outro exemplo são as diferentes descrições que um mesmo produto real pode receber em lojas de comércio eletrônico (KÖPCKE et al., 2012).

A resolução de entidades tem sido estudada por décadas (FELLEGI; SUNTER, 1969; NEWCOMBE et al., 1959). Diversas abordagens para a resolução de entidades têm sido propostas, principalmente para bases de dados cujas referências são estruturadas, como é o caso das listas de correspondência. No entanto, a resolução de entidades tem se tornado um problema ainda mais desafiador. Atualmente, a maioria dos conjuntos de dados em aplicações Web é composta por referências não estruturadas ou semiestruturadas. As referências bibliográficas em uma biblioteca digital, por exemplo, são referências semi estruturadas. Já as ofertas de produtos em uma loja de comércio eletrônico, geralmente descritas por um único texto curto, são exemplos de referências não estruturadas.

Ademais, o grande volume de dados e a constante atualização dos mesmos aumenta a complexidade do problema. Nesse caso, a resolução de entidades se torna rapidamente obsoleta e realizar todo o processo de resolução considerando também os novos dados é custoso e impraticável.

Neste trabalho é proposta uma abordagem incremental que utiliza aprendizagem de máquina para a resolução de entidades. Nas próximas seções são apresentados a contextualização

e motivação do trabalho e a proposta do mesmo, bem como sua justificativa e seus objetivos. Por fim, ao final do capítulo, é apresentada a estrutura do documento.

1.1 Contextualização

A compra de produtos por meio da Web tem se tornado algo comum na vida das pessoas por uma série de motivos, a maioria deles relacionados à comodidade proporcionada por este serviço (ZHANG; MUKHERJEE; SOETARMAN, 2013). A fim de se manterem competitivas, empresas que trabalham com comércio eletrônico têm buscado atingir seus clientes de várias formas, e para isso muitas delas têm usado dados de diferentes fontes na tomada de suas decisões.

Por exemplo, seja o cenário em que uma empresa quer usar os dados de suas vendas, mas também quer usar dados coletados em redes sociais, dados de tempo, uma base de dados com informações sobre o comportamento de compras de seus clientes e qualquer outra base de dados disponível na tomada de suas decisões. Com esses dados, é possível que tal empresa minere padrões que permitam avaliar o melhor momento para o lançamento de novos produtos, demandas em regiões particulares, estratégias de marketing personalizado e até mesmo a análise de sentimentos dos clientes a respeito de seus produtos. Nesse contexto, a chave para extrair o máximo de informações estratégicas dos dados é realizar a integração dos mesmos, transformando dados complexos em algo que pode ser mais facilmente analisado.

Nesse cenário, muitas das bases de dados são geradas por pessoas ou dispositivos eletrônicos, o que faz com que esses dados não apresentem qualquer estrutura, dificultando o processo de integração das mesmas. É comum que as referências a produtos sejam descritas por um único atributo textual curto, como por exemplo as referências “Samsung Galaxy S7” e “Smartphone Galaxy S7”. Essas duas referências devem ser identificadas como correspondências à mesma entidade, enquanto a referência “Samsung Galaxy S7 Edge” deve ser identificada como correspondência a uma entidade diferente.

Desafios similares são observados em uma biblioteca digital, que mantém informações bibliográficas obtidas automaticamente de várias fontes. Nessas bibliotecas, é comum referências distintas a um mesmo trabalho, com possíveis erros de ortografia ou mesmo erros decorrentes da coleta automatizada. Também é comum diferentes representações da lista de autores e nomes de veículos de publicações ambíguos. A Figura 1.1 ilustra um exemplo em que o *workshop* “ACM International Workshop on Web Information and Data Management” é descrito de ma-

neiras distintas, e evidencia que a padronização das citações não só melhora a qualidade dos dados mas também é fundamental para evitar confusões e má interpretação (PEREIRA; SILVA; ESMIN, 2014).

Figura 1.1 – Exemplo de referências distintas a um veículo de publicação.

- ACM International Workshop on Web Information and Data Management
- International Workshop on Web Information and Data Management
- Annual ACM International Workshop on Web Information and Data Management

Fonte: Do autor (2017).

O problema de se identificar quais das descrições de produtos em várias bases de dados se referem ao mesmo produto do mundo real é referenciado na literatura como *product matching* (KÖPCKE et al., 2012). Já a padronização de citações bibliográficas é referenciada como *citation matching* (LEE et al., 2007). Ambos são casos particulares do problema de resolução de entidades. Bons estudos sobre a resolução de entidades são apresentados por Enríquez et al. (2017), Köpcke, Thor e Rahm (2010), Elmagarmid, Ipeirotis e Verykios (2007) e Bhattacharya e Getoor (2007a).

Métodos tradicionais para a resolução de entidades comparam pares de referências com o objetivo de decidir se elas correspondem à mesma entidade. Esses métodos podem utilizar aprendizagem de máquina ou não, e quando usam a aprendizagem pode ser supervisionada ou não supervisionada. Adicionalmente, um método de *clustering* pode ser aplicado com o objetivo de agrupar as referências à cada entidade distinta.

O grande volume de dados e a constante atualização dos mesmos faz os resultados da resolução de entidades rapidamente obsoletos. Realizar todo o processo de resolução novamente usando tanto os dados já conhecidos como os novos dados leva muito tempo. Uma alternativa a esse modelo *batch* é a utilização de uma abordagem incremental, como as propostas em Gruenheid, Dong e Srivastava (2014) e Whang e Garcia-Molina (2014), que atualizam a resolução de entidades com os novos dados à medida que eles se tornam disponíveis com o passar do tempo.

Em ambientes que novos dados estão continuamente disponíveis, como no contexto de vendas *online* e de bibliotecas digitais, é comum que os conceitos conhecidos pelo algoritmo sofram alterações com o passar do tempo. Um conceito é uma função que deve ser aprendida pelo algoritmo e estabelece um mapeamento entre entradas e saídas definidas a partir de um

conjunto de exemplos (MITCHELL, 1997). Assim, uma habilidade desejável dos métodos incrementais é a capacidade de se adaptar em função dos novos dados. Para isso, o algoritmo deve ser capaz de lidar com três fenômenos, conhecidos como evolução do conceito, mudanças nos conceitos e evolução de atributos.

A evolução de conceitos (*concept evolution*) (MASUD et al., 2013) é um fenômeno caracterizado pelo surgimento de um conceito (entidade) até então desconhecido pelo algoritmo. Por exemplo, no contexto de uma loja de comércio eletrônico, a oferta de um novo produto requer que o algoritmo se atualize e seja capaz de resolver também a nova entidade. Mais interessante ainda é a capacidade de o algoritmo detectar automaticamente que um novo conceito está emergindo, e então se atualizar. Ressalta-se que em ambientes dinâmicos o número de conceitos não é conhecido nem previamente definido, e novos conceitos podem surgir a qualquer momento.

Por outro lado, as mudanças do conceito (*concept drift*) ocorrem quando conhecimentos já observados pelo algoritmo passam por alterações. Isso pode ser observado, por exemplo, quando um veículo de publicação é renomeado. Nesse caso, pode ser observado uma mudança na distribuição que gera os dados (GAMA et al., 2014), e é de suma importância que o algoritmo incremental seja capaz de tratar tais mudanças.

Finalmente, um algoritmo incremental deve ser capaz de tratar o surgimento de novos atributos (*feature evolution*), fenômeno que pode ser observado, por exemplo, quando uma acrônimo ou sigla é criado para um veículo de publicação.

1.2 Proposta deste trabalho

Tendo em vista a problemática envolvida na resolução de entidades e nos cenários que envolvem o processamento de dados em ambientes que novos dados são gerados com o passar do tempo, neste trabalho foi proposta uma abordagem incremental para a resolução de entidades. Diferentemente de outras abordagens que usam métodos de classificação para decisões binárias de casamento ou não de pares de referências e um método de *clustering* para agrupar registros da mesma entidade, neste trabalho são utilizados classificadores multiclasses (ALY, 2005) com auto treinamento e detecção de novas classes para agrupar referências à mesma entidade. Na abordagem proposta, cada classe corresponde a uma entidade distinta. O auto treinamento permite a atualização do modelo de aprendizagem na fase de teste, enquanto o me-

canismo de detecção de novas classes possibilita a identificação de referências a entidades que não ocorreram em tempo de treinamento.

A abordagem proposta é denominada AssocIER e usa algoritmos de aprendizagem de máquina supervisionados, compostos como um *ensemble*. O principal algoritmo é baseado no classificador associativo proposto por Oliveira (2014) e Oliveira e Pereira (2017), modificado para permitir sua atualização incremental. O classificador associativo utilizado usa as descrições de referências de dados para gerar um classificador, em que o modelo gerado é um conjunto de regras de associação que são utilizadas para a resolução de entidades. As regras de associação possuem a forma $X \rightarrow c_i$, onde X (antecedente da regra) é um ou mais *tokens* obtidos da descrição dos registros e c_i (consequente da regra) é o rótulo da entidade à qual o registro de dados corresponde. A geração das regras de associação foi implementada com um índice invertido (BAEZA-YATES; RIBEIRO-NETO, 2011), que pode ser facilmente modificado para permitir a atualização eficiente do modelo de aprendizagem.

Na fase de teste (ou predição) do algoritmo, um ou mais *tokens* de cada referência do incremento são comparados com os antecedentes das regras que compõem o modelo associativo, a fim de se predizer à qual entidade a referência de teste corresponde. Esse modelo foi combinado com outras técnicas de classificação baseadas em similaridade, constituindo uma abordagem baseada em *ensemble* de classificadores. Quando a referência do incremento não pode ser decidida pelo classificador associativo, o método usa as similaridades dos classificadores auxiliares para decidir se a referência corresponde a uma nova entidade ou se a predição da referência deve ser adiada, aguardando pela atualização do modelo com as próximas referências do incremento.

Não é objetivo do AssocIER tratar dados cujas referências são estruturadas ou textos longos. Cada referência de entrada para o método é uma única *string* curta (cerca de 300 caracteres). Dados cujas referências são semiestruturadas, como referências bibliográficas, podem ser utilizados, desde que os seus atributos sejam tipicamente textuais e sejam concatenados em uma única *string*.

1.3 Justificativa

Mesmo com a disponibilidade de vários trabalhos na literatura que abordam a resolução de entidades, os resultados demostram que muitas melhorias ainda podem ser propostas. Poucos trabalhos para solução do problema consideram dados cujas referências são não estruturadas

(KÖPCKE; THOR; RAHM, 2010), como pode ser observado no caso de ofertas de produtos em comércio eletrônico, no contexto de bibliotecas digitais e outros cenários em que as referências de dados são redigidas livremente sem qualquer padronização.

Poucos trabalhos consideram a resolução de entidades em ambientes que novos dados estão disponíveis com o passar do tempo, e portanto, demanda por uma abordagem incremental. Gruenheid, Dong e Srivastava (2014), Whang e Garcia-Molina (2014) e Welch, Sane e Drome (2012) propõem abordagens para a resolução de entidades de forma incremental que consideram, inclusive, a evolução e mudanças no conceito, mas são aplicadas na resolução de entidades definidas por dados estruturados. Bilenko, Basil e Sahami (2005) tratam a resolução de entidades descritas por dados cujas referências são estruturadas sob uma perspectiva incremental, mas não realizam a detecção de novas entidades. Ferreira et al. (2010), Ferreira et al. (2014) e Santana et al. (2015) tratam a resolução de entidades cujas referências são descritas por dados semiestruturados de forma incremental e inclusive tratam o surgimento de novas entidades, mas para o caso particular de resolução de entidades que desambigua nomes de autores em citações bibliográficas, e portanto, exploram características particulares desse cenário.

Como pode ser observado, a resolução de entidades de forma incremental é um problema recente, que ainda demanda pesquisa e propostas de novas abordagens para a sua solução. Muitos desafios decorrentes da era *Big Data* ainda precisam ser superados, de forma que autores reconhecidos na comunidade de banco de dados e gigantes da tecnologia também têm investigado o problema, cada um com propostas de abordagens para dados com características específicas.

Dada a importância do problema, este trabalho teve como foco o desenvolvimento de soluções para a resolução de entidades descritas por dados textuais curtos de maneira incremental, sobretudo quando referências a novas entidades surgem entre os novos dados.

1.4 Objetivo geral e específicos

O objetivo geral deste trabalho foi o desenvolvimento de uma abordagem incremental para a resolução de entidades descritas por dados textuais curtos, em ambientes que novas referências de dados são geradas com o passar do tempo. Para isso, estabeleceram-se os seguintes objetivos específicos:

a) adaptar o classificador associativo proposto por Oliveira (2014) e Oliveira e Pereira (2017). Essa atualização consiste na adição ou remoção de novas regras de associação geradas a partir das descrições das referências do conjunto de teste;

b) atualizar os demais modelos usados para a composição do *ensemble* com informações obtidas em referências de dados do incremento;

c) desenvolver mecanismos para a detecção e processamento de entidades que não ocorreram na fase de treino e que surgiram na fase de teste;

d) avaliar classificadores para a formação do *ensemble* proposto;

e) propor um ambiente de experimentação usando várias bases de dados reais, principalmente relacionadas ao cenário de vendas *online* e bibliotecas digitais;

f) avaliar os resultados obtidos pelo método e compará-los com *baselines*;

g) submissão de um artigo a um periódico do índice restrito do Qualis Capes.

1.5 Contribuições deste trabalho

As principais contribuições deste trabalho são:

a) proposta e implementação do método AssocIER para a resolução de entidades descritas por dados não estruturados de forma incremental. No melhor de nosso conhecimento, este é o primeiro trabalho a utilizar classificação multiclasses para a resolução de entidades;

b) avaliação do método proposto através de experimentação e comparação *baselines*;

c) implementação da abordagem para resolução de entidades tradicional não incremental baseada em aprendizagem de máquina descrita por Köpcke, Thor e Rahm (2010) e usada como *baseline*;

d) desenvolvimento de uma ferramenta para suporte à analise de resultados, disponível em https://joaoasilva.shinyapps.io/resultsApp/;

e) submissão do artigo “A Multiclass Classification Approach for Incremental Entity Resolution” em um periódico do índice restrito do Qualis Capes.

1.6 Metodologia

Segundo Prodanov e Freitas (2013), metodologia é um conjunto de métodos, técnicas e procedimentos que viabilizam a execução da pesquisa e geram como resultado um novo produto, processo ou conhecimento. Nas próximas seções, o trabalho é classificado em relação ao tipo de pesquisa e a metodologia executada é descrita.

1.6.1 Tipo de pesquisa

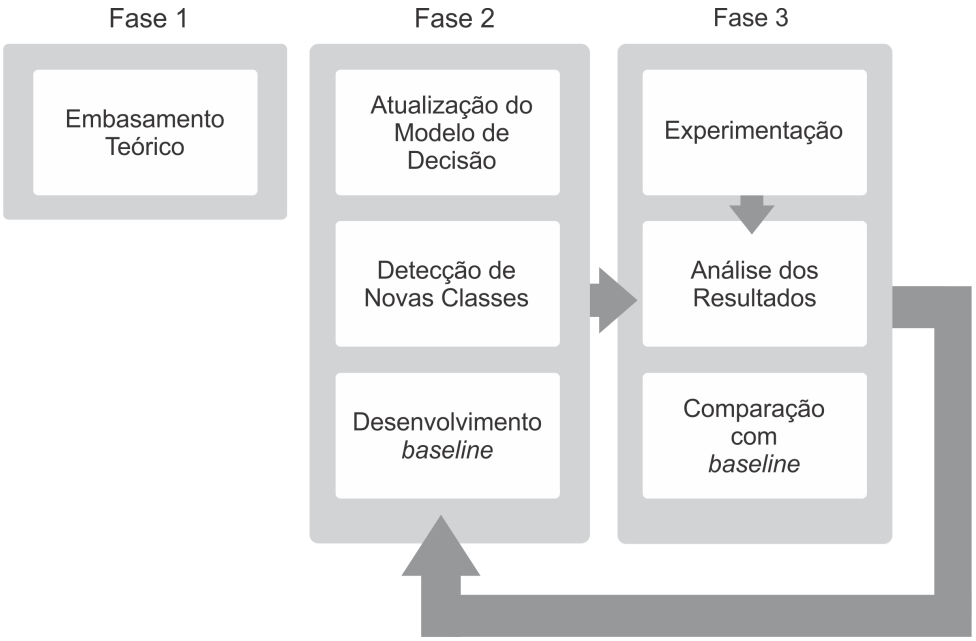
Uma pesquisa pode ser classificada quanto à sua natureza, quanto aos seus objetivos, quanto à sua abordagem e quanto aos seus procedimentos (JUNG, 2004) . A pesquisa desenvolvida neste trabalho pode ser classificada como aplicada quanto à sua natureza, isto é, a pesquisa é fundamentada em *design science*, uma vez que utiliza conhecimentos já estabelecidos para gerar novos conhecimentos com fins de aplicação, através do desenvolvimento de ferramentas necessárias para ações adequadas no domínio dos profissionais em seus campos de atuação. É uma pesquisa exploratória quanto a seu objetivo, pois promove a familiaridade com o problema. É uma pesquisa quantitativa quanto à abordagem do problema, pois os resultados obtidos são avaliados através de um conjunto de métricas que traduzem em números os conhecimentos gerados. Quanto aos procedimentos, é uma pesquisa experimental, uma vez que os parâmetros definidos influenciam nos resultados obtidos.

1.6.2 Método de pesquisa

Este trabalho foi executado de março de 2015 a fevereiro de 2017, e seguiu as fases observadas na Figura 1.2: embasamento teórico, implementação de abordagens incrementais para a resolução de entidades utilizando aprendizagem de máquina e desenvolvimento do *baseline*, experimentação e avaliação dos resultados.

Embasamento teórico: Nesta fase, foram estudados os principais trabalhos sobre resolução de entidades e abordagens utilizadas em sua solução. Uma vez definidas essas abordagens, foi feito um referencial teórico sobre o mesmo, compreendendo os seguintes conceitos: resolução de entidades, funções de similaridade e aprendizagem de máquina. Também foram estudadas as técnicas de aprendizagem incremental, a teoria envolvida nos fenômenos de mudança e evolução do conceito e outros. O referencial teórico desenvolvido é apresentado no Capítulo 2. Adicionalmente, foram realizados estudos dos trabalhos propostos por (PEREIRA; SILVA; ESMIN, 2014), Oliveira e Pereira (2017) e Oliveira (2014), e trabalhos relacionados às

Figura 1.2 – Método de pesquisa.



Fonte: Do autor (2017).

abordagens incrementais para a resolução de entidades. O objetivo foi a familiarização com o problema em estudo e as abordagens utilizadas em sua solução.

Implementação de abordagens incrementais para a resolução de entidades e do *baseline*: Nesta fase do projeto foram implementadas soluções baseadas em aprendizagem de máquina capazes de realizar a resolução de entidades. Primeiramente, foi implementada a atualização do classificador associativo proposto por Oliveira e Pereira (2017) e Oliveira (2014) e mecanismos para a detecção de novas classes. Nesta fase também foi implementada a abordagem não incremental discutida por Köpcke, Thor e Rahm (2010), usada como *baseline* na comparação de resultados.

Experimentação e avaliação dos resultados: Nesta fase, as abordagens implementadas na fase anterior foram experimentadas como descritas no Capítulo 4. Os resultados obtidos foram analisados a fim de determinar a qualidade dos resultados obtidos, no que diz respeito a sua eficiência e eficácia.

As fases implementação de abordagens e experimentação e avaliação de resultados foram repetidas várias vezes, buscando investigar abordagens que obtivessem resultados satisfatórios. O embasamento teórico foi expandido sempre que necessário nessas fases.

1.7 Estrutura do documento

O restante deste documento está estruturado da seguinte forma. Um referencial teórico sobre conceitos importantes para a realização do trabalho é apresentado no Capítulo 2, abordando conceitos como funções de similaridades, aprendizagem de máquina e o problema de resolução de entidades e também trabalhos correlacionados ao projeto. A abordagem incremental proposta para a resolução de entidades é apresentada no Capítulo 3. A configuração do ambiente experimental é apresentada no Capítulo 4. Os resultados obtidos pelo método e sua avaliação são apresentados no Capítulo 5. No Capítulo 6 são apresentadas as conclusões obtidas e direcionamentos para trabalhos futuros. Por fim, é apresentada a bibliografia consultada para a realização do projeto.

2 REFERENCIAL TEÓRICO

Nesta seção estão descritos conceitos sobre o problema de resolução de entidades, funções de similaridade, aprendizagem de máquina, classificação de dados, regras de associação e classificação em fluxos contínuos de dados e outros conceitos necessários para a compreensão da proposta apresentada.

2.1 Resolução de entidades

Bases de dados contêm registros que podem se referir a uma mesma entidade do mundo real, sem necessariamente serem idênticas. Segundo (BENJELLOUN et al., 2009), várias representações de uma mesma entidade real podem surgir, causadas por erros de digitação, abreviações ou mesmo no processo de integração de duas ou mais bases de dados. Um caso particular em que esse problema ocorre é verificado em dados recuperados da Web, referências são não estruturadas ou semiestruturadas são extraídas automaticamente. Assim, a Resolução de Entidades tem como objetivo identificar grupos de referências em uma base de dados que correspondem a uma mesma entidade no mundo real. O problema de resolução de entidades foi introduzido por NewCombe et al. (1959) e formalizado 10 anos mais tarde por Fellegi e Sunter (1969).

A dificuldade e a importância do problema de resolução de entidades e suas variações têm recebido grande atenção em diversas pesquisas em várias áreas do conhecimento. Bons estudos sobre a resolução de entidades são encontrados em Enríquez et al. (2017), Getoor e Machanavajjhala (2012), Köpcke, Thor e Rahm (2010) e Elmagarmid, Ipeirotis e Verykios (2007).

2.1.1 Formulação do problema

Seja uma base de dados $I = \{d_1, d_2, ..., d_n\}$, em que cada cada registo d_i em I é um registro estruturado $F = \{f_1, f_2, ..., f_m\}$, onde f_q é um atributo específico do domínio. Seja ainda um conjunto de entidades do mundo real $\mathscr{E} = \{e_1, e_2, ..., e_p\}$ em que o número de entidades p geralmente não é conhecido e nem fornecido como entrada para o algoritmo. Dois registros d_i e d_j são ditos co-referências da entidade e_k, denotado por $d_i \sim d_j$, se ambos os registros fazem referência a uma mesma entidade, para qualquer $i \neq j$. O conjunto G_l para e_k é o conjunto de todos os registros $d_i \in I$ que fazem referência à mesma entidade e_k, isto é, $G_l = \{d_i, d_j \in I : d_j \sim d_i\}$. O conjunto G_l também não é conhecido pelo algoritmo, e assim o problema de resolução

de entidades consiste em identificar o conjunto de entidades $\mathscr{E}$ com as quais os registros de I se relacionam e também o conjunto G_l para cada entidade e_k.

Em outras palavras, o objetivo de um algoritmo de resolução de entidades é subdividir a base de dados I em um conjunto de grupos $\mathscr{G} = \{G_1, G_2, ..., G_p\}$, de modo que todas as referências em cada partição G_k correspondam à mesma entidade k (alta precisão) e não mais que uma partição contenha referências a tal entidade (alta revocação). Se dois registros d_i e d_j são alocados à mesma partição G_k, então o algoritmo acredita que esses dois registros são co-referências, isto é, $d_i \sim d_j$. O conjunto de partições $\mathscr{G}$ é dito correto se $A(e_k) = G_l$ para todas os registros $d_i \in I$. $A(e_k)$ é uma função que mapeia um registro d_i em uma partição $G_l \in \mathscr{G}$ correspondente. O algoritmo de resolução de entidades que gera os agrupamentos de $\mathscr{G}$ a partir da base de dados I é denotado por ER, isto é, $ER(I) = \mathscr{G}$.

Seja a operação *insert*, caracterizada pela adição de um novo registro d_i à base de dados I, a operação *delete*, caracterizada pela remoção de um registro de d_i de I e a operação *change*, caracterizada pela modificação de um ou mais atributos de um registro $d_i \in I$. O conjunto dessas operações de atualização realizadas em algum momento do tempo é chamado **incremento**, denotado por ΔI, de forma que aplicação do incremento ΔI a I pode ser denotado por $\Delta I + I$. Como as operações em um incremento podem ser do tipo *delete* ou *change*, o número de registros após a aplicação do incremento ΔI depende do número de registros originais em I e do número de registros em ΔI, ou seja, $|I + \Delta I| \leq |I| + |\Delta I|$. Assim, a resolução de entidades incremental pode ser definida como a aplicação do incremento ΔI à base de dados I, de modo que os agrupamentos em $\mathscr{G}$ sejam atualizados após a aplicação do incremento. O método incremental será denotado por IER e o resultado da aplicação do método por $IER(I, \Delta I, \mathscr{G})$.

Para (GRUENHEID; DONG; SRIVASTAVA, 2014), são dois os objetivos de um método incremental para resolução de entidades: **i)** o método incremental deve ser muito mais rápido que um método em *batch*, especialmente quando o número de operações no incremento é pequeno. Isto é, aplicar $IER(I, \Delta I, \mathscr{G})$ deve ser muito mais rápido que aplicar $ER(I + \Delta I)$, quando $|\Delta I| \ll |I|$; e **ii)** o método incremental deve alcançar resultados similares ao método *batch*, isto é, $IER(I, \Delta I, \mathscr{G}) \approx ER(I + \Delta I)$, em que $\approx$ denota agrupamentos com precisão e revocação similares.

Uma operação *change* em um incremento caracteriza uma mudança na distribuição que gera os dados, e a função $A(e_k)$ que mapeia um registro d_i em uma partição G_l precisa ser atualizada. Essa mudança do relacionamento entre os atributos do registro d_i e a variável alvo

G_l é um fenômeno conhecido na literatura como mudanças do conceito (*concept drift*) (GAMA et al., 2014). Em outras palavras, a mudança do conceito ocorre quando pode ser observada uma mudança nas definições das entidades e_k ao longo do tempo. De acordo com (GAMA et al., 2014), a mudança de conceito pode ser real, se a associação entre os atributos e a variável alvo muda, ou virtual, caso em que os atributos que descrevem os dados sofrem alterações ao longo do tempo. Note que como a operação *change* pode ser alcançada através da remoção de um registro e a adição de um novo (GRUENHEID; DONG; SRIVASTAVA, 2014), as operações *delete* e *insert* também podem levar a mudanças de conceito.

Por outo lado, se a aplicação de um incremento ΔI a I levar à adição de um ou mais atributos a F, é observado um fenômeno conhecido como evolução dos atributos (*feature evolution*) (MASUD et al., 2013), e nesse caso a função $A(e_k)$ também precisa se adaptar para utilizar os novos atributos no mapeamento de novo registro d_i em uma partição G_l. Finalmente, se as operações *insert* em um incremento levarem à criação de um novo grupo G_l em $\mathscr{G}$, isto é, referências a uma nova entidade e_k passaram a ocorrer nos novos dados, o fenômeno é conhecido como evolução do conceito (*concept evolution*) (MASUD et al., 2013), e o algoritmo deve ser capaz de se adaptar para permitir o mapeamento da nova entidade.

Métodos incrementais para a solução do problema têm o objetivo de manter a resolução de entidades atualizada sempre que novos dados estão disponíveis. A ideia principal de tais métodos é não realizar toda a resolução de entidades novamente considerando tanto dados antigos quantos os novos dados, apenas atualizar uma solução já existente com as informações dos novos dados (RAMADAN et al., 2015). Assim, o tratamento dos fenômenos mudança dos conceitos, evolução dos atributos e evolução dos conceitos são características importantes para um método incremental para solução do problema.

2.1.2 Soluções para a resolução de entidades

Um processo para a resolução de entidades geralmente considera uma série de transformações nos dados (TALBURT, 2010; GALHARDAS et al., 2001) conforme descritas a seguir:

a) **preparação dos dados:** consiste em transformar os dados em um formato mais conveniente para o processo de resolução. Essa fase ainda pode envolver a transformação dos dados em um conjunto de atributos (*features*), a determinação dos tipos de cada atributo, a normalização de valores e outros pré-processamentos nos dados, como a remoção de *stopwords* (preposições, conjunções, artigos...), sinais de pontuação e outros;

b) ***blocking*:** divide a base de dados em subconjuntos (blocos) de modo que **i)** se dois registros podem fazer referência à mesma entidade, então eles são colocados juntos em pelo menos um bloco e **ii)** se dois registros não estão no mesmo bloco, é provável que eles não façam referência à mesma entidade (BAXTER; CHRISTEN, 2003; MCCALLUM; NIGAM; UNGAR, 2000). O objetivo do *blocking* é deixar o algoritmo de resolução de entidades mais eficiente. Em métodos tradicionais, os algoritmos tem custo quadrático em função do número de registros, já que todos os pares de registros precisam ser comparados. Com o *blocking*, apenas pares de registros do mesmo bloco são comparados, reduzindo significativamente o tempo de execução do algoritmo;

c) **representação do problema:** consiste em estabelecer um mapeamento entre a base de dados e o modelo utilizado no algoritmo para resolução de entidades. Por exemplo, em Gruenheid, Dong e Srivastava (2014), os autores utilizam um grafo de similaridades para agrupar referências à mesma entidade. Para isso os autores consideram cada registro da base de dados como um vértice e uma aresta entre dois vértices representa a similaridade entre dois registros;

d) **computação das similaridades:** computam a similaridade entre cada par de registros. Essa fase geralmente é computacionalmente cara, dada a necessidade de comparar todos os pares de registros da base de dados. Mesmo com a utilização de técnicas de *blocking* para a redução de pares a serem comparados, o cálculo da similaridade ainda é caro, já que muitas vezes é necessário consultar fontes de informações externas ou considerar alguma interação humana através de *crowdsourcing* (VESDAPUNT; BELLARE; DALVI, 2014; NURAY-TURAN; KALASHNIKOV; MEHROTRA, 2012);

e) **agrupamento dos registros:** esta fase do processo é responsável por agrupar todos os registros que fazem referência à mesma entidade, e geralmente é realizada com um algoritmo de aprendizagem de máquina não supervisionado (*clustering*) (NURAY-TURAN; KALASHNIKOV; MEHROTRA, 2012; CHAUDHURI; GANTI; MOTWANI, 2005);

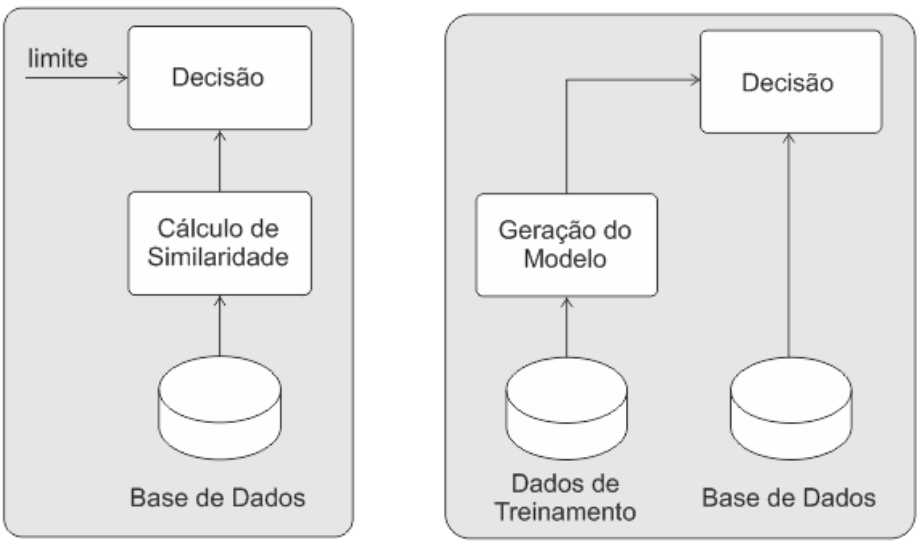
f) ***merging*:** tem o objetivo de agrupar os *clusters* da fase anterior em um único *cluster* mais representativo (BENJELLOUN et al., 2009; HERZOG; SCHEUREN; WINKLER, 2007).

Para Köpcke, Thor e Rahm (2010), uma abordagem para resolução de entidades pode ser baseada em aprendizagem de máquina ou não. As abordagens sem aprendizagem de máquina

(Figura 2.1a) utilizam a similaridade entre duas referências, isto é, verificam se a similaridade $sim(d_i, d_j)$ entre duas referências d_i e d_j é superior a algum limite estabelecido. Em caso positivo, essas duas referências são consideradas referências a uma mesma entidade. Por outro lado, nas abordagens que utilizam aprendizagem de máquina (Figura 2.1b), a resolução de entidades pode ser considerada como um problema de classificação, em que para todos os pares de referências (d_i, d_j) de duas fontes de dados R e S, o classificador deve determinar se (d_i, d_j) faz referência à mesma entidade ou não (KOLB et al., 2011). Na Figura 2.2a é apresentado um exemplo de execução de uma abordagem para resolução de entidades sem a utilização de aprendizagem de máquina. Nesse caso, os pares de referências são comparados através de uma função de similaridade, e se a semelhança entre elas for superior a um limiar estabelecido, os registros são considerados referências à mesma entidade. Já a Figura 2.2b ilustra um exemplo de abordagem com utilização de aprendizagem de máquina, em que pares de referências são usados para treinamento de um classificador binário, que na fase de teste deve decidir se os registros de um novo par fazem referência à mesma entidade ou não.

Algumas abordagens também utilizam a Web como fonte de informação adicional (PEREIRA et al., 2011), (ELMACIOGLU et al., 2007). Nessas últimas, informações das entidades são submetidas como consultas a uma máquina de busca Web. O conjunto resposta para as con-

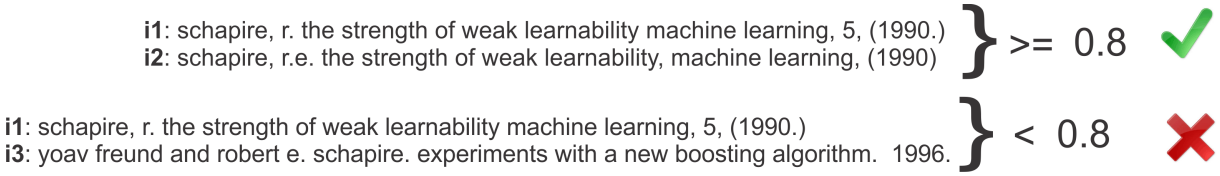
Figura 2.1 – Abordagens para a identificação de registros de dados que correspondem à mesma entidade: (a) sem aprendizagem de máquina e (b) com aprendizagem de máquina. Adaptado de Köpcke, Thor e Rahm (2010).



(a) (b)

Fonte: Do autor (2017).

Figura 2.2 – Diferentes abordagens para a resolução de entidades.



(a) Resolução de entidades sem a utilização de aprendizagem de máquina.

(b) Resolução de entidades com utilização de aprendizagem de máquina.

Fonte: Do autor (2017).

sultas são então analisadas a fim de se extrair informações que são utilizadas para a resolução de entidades.

Diversas abordagens para a resolução de entidades têm sido propostas, principalmente para bases de dados cujas referências são estruturadas. Na Figura 2.3a são apresentados exemplos de referências a dados estruturados, isto é, as referências aos dados são formadas por atributos precisamente definidos. Em abordagens tradicionais para a resolução de entidades, é realizada a comparação dos registros em um par de referências. Uma ou mais funções de similaridades são aplicadas aos atributos das referências com o objetivo de calcular o quão semelhantes as referências dos pares são e, então, uma estratégia é utiliza para decidir se as referências no par correspondem à mesma entidade ou não. Para esse tipo de dados, é comum a utilização de uma mesma função de similaridade em todos os atributos ou funções distintas, de acordo com as características de cada campo que compõem o referência estruturada.

As referências na maiorias das bases de dados são semiestruturadas ou não estruturadas, como os exemplos da Figura 2.3b e 2.3c, respectivamente. Em referências semiestruturadas,

a definição de atributos existe, mas eles não são precisamente restritos. Em referências não estruturadas, por outro lado, nenhum padrão de atributos pode ser observado.

Figura 2.3 – Exemplos de referências estruturadas, semiestruturadas e não estruturadas.

Nome	Endereço	E-mail	Telefone
João A. Silva	R. 3, Centro - Lavras	joaoasilva@mail.com	(35) 9999-9999
Pedro H. Silva	R. dos Ipês, Bairro Boa Vista - Alfenas	phs@mail.com	(35) 8888-0000
João Antonio Silva	R. 3, Centro - Lavras	joaoasilva@gmail.com	(35) 9999-8888

(a) Referências estruturadas.

P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire,	Gambling in a Rigged Casino: The Adversarial Multi-armed Bandit Problem	In proc. 36th Annual Symposium on Foundations of Computer Science	
A. Blum, M. Furst, M. J. Kearns, and Richard J. Lipton.	Cryptographic Primitives Based on Hard Learning Problems	In Pre-proceedings of Crypto '93	Pages 1-24
Harris Drucker, Robert Schapire, and Patrice Simard	Improving Performance in Neural Networks Using a Boosting Algorithm	In Advances in Neural Information Processing Systems 5	

(b) Referências semiestruturadas.

Samsung Galaxy S7
Smartphone Galaxy S7
Samsung Galaxy S7 Edge

(c) Referências não estruturadas.

Fonte: Do autor (2017).

2.2 Funções de similaridade

Uma forma de comparar o quanto referências em um conjunto de dados são semelhantes consiste na utilização de funções de similaridade, isto é, a realização uma comparação entre atributos dessas referências. No caso em que as referências são constituídas por atributos do tipo *string* (cadeia de caracteres), as funções de similaridade mapeiam a semelhança entre duas *strings* str_i e str_j em um número real sim, $0 \leq sim \leq 1$. Valores de sim próximos a 0 indicam baixa similaridade enquanto valores próximos de 1 indicam alta similaridade.

Diversas funções de similaridade para *strings* são encontradas na literatura, geralmente divididas em dois grupos. As funções baseadas em caracteres determinam a similaridade entre duas *strings* a partir do número de operações necessárias para se transformar uma *string* str_i em uma *string* str_j. Já nas funções baseadas em *tokens* as *strings* são transformadas em *tokens* (palavras), que são utilizados para o cálculo da similaridade (PEREIRA, 2009).

2.2.1 Funções de similaridade baseadas em caracteres

Várias funções de similaridade baseadas em caracteres foram propostas, tais como a Jaro e a Distância de Edição ou Distância de Levenshtein.

Distância de Edição: A Distância de Edição (LEVENSHTEIN, 1966) é a mais conhecida e utilizada das funções de similaridade baseadas em caracteres, sendo calculada a partir do número de operações necessárias para transformar uma *string* em outra. Essas operações consistem em inserções, substituições ou eliminações de caracteres. Dadas duas *strings* str_i e str_j, a distância de edição entre estas strings é dada pela Equação(2.1).

$$edSim(str_i, str_j) = 1 - \frac{ed(str_i, str_j)}{mim(|str_i|, |str_j|)} \tag{2.1}$$

Em que $mim(|str_i|, |str_j|)$ é o comprimento da menor *string* e $ed(str_i, str_j)$ o número de operações necessárias para transformar str_i em str_j.

Métrica Jaro: Outra métrica de similaridade entre *strings* é a métrica Jaro (JARO, 1989), que considera o número e a ordem de caracteres comuns de duas *strings*. Sejam $str_i = a_1 ... a_K$ e $str_j = b_1 ... b_L$. Seja também a_k em str_i um caracter comum em str_j, tal que $b_l = a_k$, para $k - H \leq l \leq k + H$ em que $H = mim(|s_i|, |s_j|)/2$. Considere ainda $str'_i = a'_1 ... a'_K$ os caracteres em str_i comuns com str_j, na mesma ordem em que aparecem em str_i, e $str'_j = b'_1 ... b'_L$ de forma análoga. A transposição de str'_i, str'_j é definida como a posição k, tal que $a'_k \neq b'_k$. Se $T_{str'_i, str'_j}$ é a metade do número de tranposições para str_i e str_j, então a similaridade de Jaro para as duas strings é dada por:

$$jaro(str_i, str_j) = \frac{1}{3} \times \left(\frac{|str'_i|}{|str_i|} + \frac{|str'_j|}{|str_j|} + \frac{|str'_i| - T_{str'_i, str'_j}}{str'_i} \right) \tag{2.2}$$

Uma importante variação da similaridade Jaro é a similaridade Jaro-Winkler (WINKLER, 1999), que dá enfase aos primeiros caracteres das *strings*.

2.2.2 Funções de similaridade baseadas em *tokens*

Nos casos em que a ordem das palavras não é importante ou em que a *string* é suficientemente longa, pode ser interessante converter as *strings* str_i e str_j em conjuntos de *tokens* (palavras) e aplicar o cálculo de similaridade sobre estes conjuntos. Dois exemplos de métrica de similaridade efetivos e muito utilizados são a similaridade de Jaccard e a similaridade do Cosseno .

Similaridade de Jaccard: O coeficiente de similaridade de Jaccard (JACCARD, 1901) quantifica a semelhança entre duas strings através da comparação de seus conjuntos de *tokens*. Para duas *strings* str_i e str_j, a similaridade de Jaccard é definida como:

$$Jaccard(str_i, str_j) = \frac{|str_i \bigcap str_j|}{|str_i \bigcup str_j|} \tag{2.3}$$

Em que $|str_i \bigcap str_j|$ é a cardinalidade da intersecção entre os conjuntos de *tokens* das *strings* e $|str_i \bigcup str_j|$ é a cardinalidade da união dos conjuntos de *tokens* das *strings.*

Similaridade do Cosseno: A similaridade do cosseno é uma função de similaridade baseada em *tokens* e mede a semelhança entre duas *strings* a partir de suas representações vetoriais (JONES, 1972), (SALTON; MCGILL, 1983). Dada uma coleção $Str = \{str_1, str_2, ..., str_{|Str|}\}$ composta por $|Str|$ *strings* e uma coleção $TK = \{tk_1, tk_2, ..., tk_{|TK|}\}$ de $|TK|$ *tokens* distintos observados na coleção Str, então cada par (tk_i, str_j), $tk_i \in TK$ e $str_j \in Str$ recebe um peso $w_{i,j} \geq 0$ indicando a importância do *token* tk_i na descrição semântica da *string* str_j, de forma que a *string* str_j seja representada como um vetor de pesos $\vec{str}_j = \{w_{1,j}, w_{2,j}, ..., w_{tk,j}\}$.

O peso associado ao *token* $w_{i,j}$ pode ser um indicativo de sua ocorrência na *string* e sua frequência na coleção, sendo o esquema de pesos *Term Frequency x Inverse Document Frequency* (TF-IDF) o mais conhecido. Neste esquema, o TF indica o número de ocorrências de um termo em uma *string* e o IDF representa a raridade do termo na coleção. O peso $w_{i,j}$ pode ser calculado pela Equação 2.4 (BAEZA-YATES; RIBEIRO-NETO, 2011).

$$w_{i,j} = \left(1 + log f_{i,j}\right) \times log \frac{N}{n_i} \tag{2.4}$$

Em que $f_{i,j}$ é a frequência do *token* k_i na *string* str_j, n_i é o número de *strings* pertencentes a S que contém o *token* tk_i e N é o total de *strings* na coleção.

A similaridade do cosseno entre duas *strings* str_i e str_j representadas em um espaço vetorial pode ser calculada como o cosseno do ângulo entre os respectivos vetores pela Equação 2.5.

$$sim(str_i, str_j) = \frac{\Sigma_{tk=1}^{|TK|} w_{tk,i} \times w_{tk,j}}{\sqrt{\Sigma_{tk=1}^{|TK|} w_{tk,i}^2} \times \sqrt{\Sigma_{tk=1}^{|TK|} w_{tk,j}^2}} \tag{2.5}$$

2.2.3 Funções de similaridade mistas

Há ainda funções de similaridade que são combinações das similaridades baseadas em *tokens* com as similaridades baseadas em caracteres, tais como a similaridade *Soft TF-IDF* (BILENKO; MOONEY, 2003), a similaridade de Segundo Nível (MONGE; ELKAN, 1996). Uma função que combina a similaridade de Jaccard e a Distância de edição é descrita por French, Powell e Schulman (2000).

Por fim vale ressaltar que o cálculo de similaridade deve ser realizado de forma cuidadosa, considerando os atributos da base de dados considerada. Enquanto as funções de similaridade baseadas em caracteres são recomendadas para *strings* curtas, as medidas que transformam a *string* em *tokens* são indicadas para *strings* mais longas (BILENKO et al., 2003).

2.3 Aprendizagem de máquina

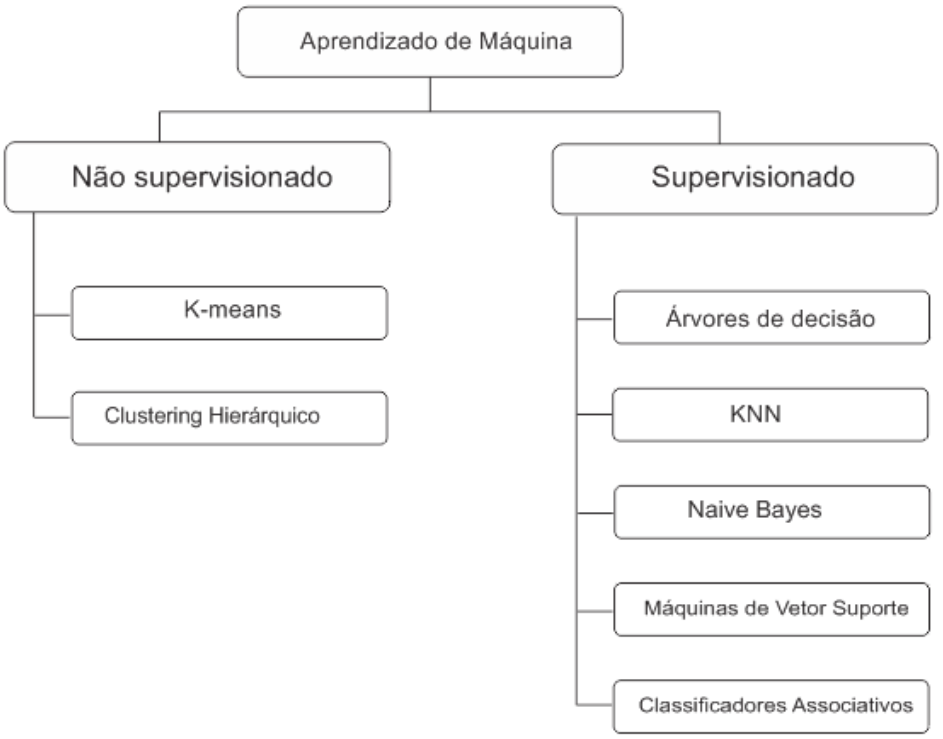
A aprendizagem de máquina é uma subárea da Inteligência Artificial que trata da proposta de algoritmos que melhorem seu desempenho de acordo com sua experiência (MITCHELL, 1997). Esses algoritmos são dependentes de uma fase de aprendizagem, na qual é gerado um modelo de decisão, que posteriormente é utilizado para realizar a predição de novos registros.

Os algoritmos de aprendizagem de máquina podem apresentar aprendizagem supervisionada ou não supervisionada (BAEZA-YATES; RIBEIRO-NETO, 2011). Enquanto para os algoritmos supervisionados são necessários dados rotulados para a geração do modelo, esses dados não são necessários para os algoritmos não supervisionados. Em outras palavras, na aprendizagem supervisionada é gerado um classificador que é capaz de determinar a classe de novos registros a partir de exemplos de treinamento pertencentes a categorias conhecidas e pré-definidas (TAN; STEINBACH; KUMAR, 2009). Já na aprendizagem não supervisionada, em que os principais representantes são os algoritmos de *clustering*, os algoritmos tentam determi-

nar se os dados podem ser agrupados de alguma maneira, de forma que maximize a semelhança de registros pertencentes a um mesmo *cluster* e minimize a semelhança entre registros de *clusters* distintos.

Na Figura 2.4 podem ser visualizados alguns dos algoritmos de aprendizagem de máquina mais conhecidos e utilizados. O maior interesse deste trabalho são os algoritmos de classificação, portanto a aprendizagem de máquina supervisionada é melhor detalhada nas próximas seções.

Figura 2.4 – Exemplos de algoritmos de aprendizagem de máquina.



Fonte: Do autor (2017).

Os algoritmos de aprendizagem ainda podem apresentar uma abordagem de aprendizagem semi-supervisionada ou híbrida, se combinarem um pequeno conjunto de dados rotulados com um grande conjunto de dados sem rótulos em busca de melhorias nos resultados (BAEZA-YATES; RIBEIRO-NETO, 2011).

Independente do tipo de aprendizado utilizado, algoritmos de aprendizagem de máquina são usados extensivamente em diversas aplicações, como no processamento de linguagem natural, diagnósticos médicos, visão computacional e recuperação de informação. Ressalta-se que, independente do problema em que que são aplicados, a efetividade aprendizagem de máquina

para a solução de um problema depende de um conjunto de dados de treinamento suficiente para desenvolver o aprendizado (KÖPCKE; THOR; RAHM, 2010).

2.4 Classificação de dados

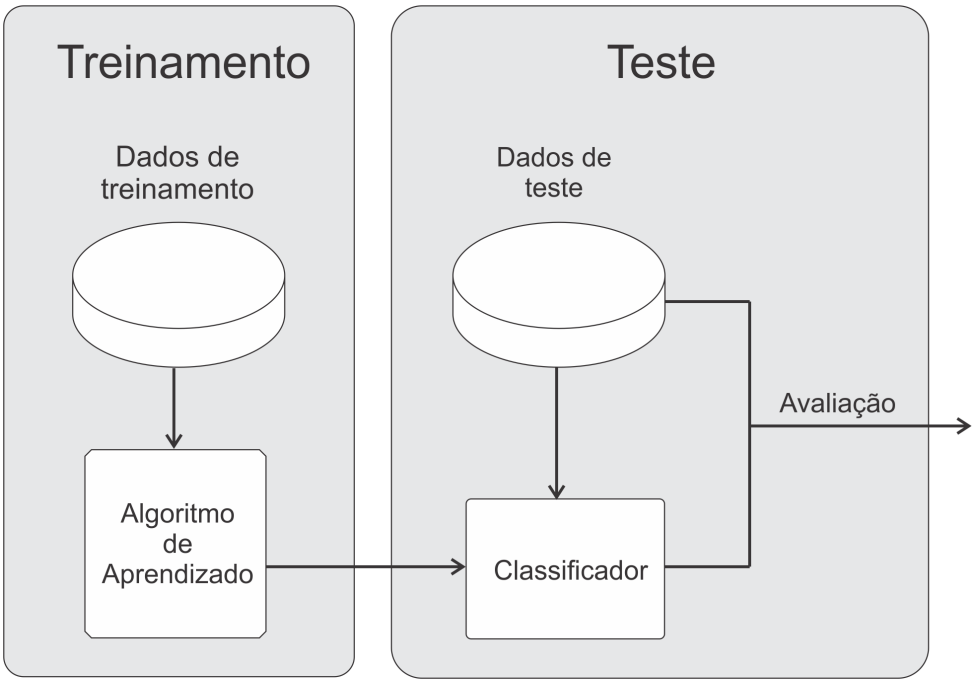
Classificação de dados é a tarefa de organizar um conjunto de registros em categorias já conhecidas e definidas. Em outras palavras, a classificação tem o objetivo de aplicar rótulos conhecidos que descrevem as classes ou categorias desses dados (BAEZA-YATES; RIBEIRO-NETO, 2011). Com o crescente número de dados armazenados digitalmente e consequente necessidade de organização, um grande esforço tem sido direcionado para a proposta de algoritmos que realizem a classificação de dados automaticamente, muitos deles utilizando algoritmos de aprendizagem de máquina.

O conjunto de dados de entrada para uma tarefa de classificação é uma coleção de registros ou exemplos, que podem ser descritos por um dupla (F, y), em que $F = \{f_1, f_2, ..., f_n\}$ é um conjunto de atributos ou *features* que descrevem o dado e y é um atributo especial, denominado rótulo ou atributo alvo, que representa a classe de um dado. Segundo Tan, Steinbach e Kumar (2009), o conjunto de atributos do conjunto F em uma tarefa de classificação pode assumir valores discretos ou contínuos, enquanto a variável alvo y deve assumir apenas valores discretos, sendo esta a característica chave na distinção entre uma tarefa de classificação e uma tarefa de regressão. Assim, a classificação pode ser definida como a tarefa de aprender uma função alvo, ou modelo de classificação, que mapeie cada conjunto de atributos F de um registro em um rótulo de classe y pré-definido.

A classificação de dados é um processo dividido em duas fases. Na primeira fase, um conjunto de dados rotulados, denominado conjunto de treinamento, é utilizado para se gerar o modelo de classificação. Nessa fase, o algoritmo aprende os padrões que descrevem as classes definidas pelos rótulos. Na segunda fase, o modelo gerado na fase de treinamento é aplicado a um conjunto de dados de teste, composto por registros que não fazem parte dos dados de treinamento. Essa fase da classificação é subdividida em duas etapas. Na primeira etapa o modelo de classificação atribui uma classe a cada registro dos dados de teste e na segunda, a atribuição do classificador é comparada com as classes definidas por especialistas humanos (BAEZA-YATES; RIBEIRO-NETO, 2011). Se a porcentagem dos dados do conjunto de teste classificados corretamente pelo classificador é aceitável, então o modelo de classificação pode

ser aplicado na predição de classes de dados não rotulados. Este processo pode ser observado na Figura 2.5.

Figura 2.5 – Processo de classificação.



Fonte: Do autor (2017).

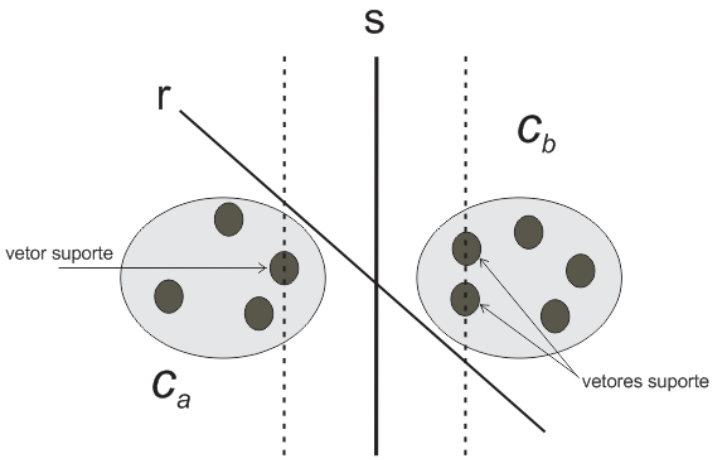
Diversos algoritmos de aprendizagem de máquina são capazes de realizar a classificação automática de dados, cada um deles com uma técnica diferente e com o mesmo objetivo de encontrar o modelo que melhor associe o rótulo dos dados de entrada a um conjunto de atributos. Nas próximas seções, são discutidas algumas dessas abordagens, mas ressalta-se que várias outras técnicas de classificação podem ser verificadas na literatura, entre os quais pode-se citar os classificadores baseados em regras, as redes neurais artificiais, o Naive Bayes, as árvores de decisão e o *k-Nearest Neighbor* (KNN). O SVM e os classificadores associativos são descritos nas próximas seções.

2.4.1 SVM

O SVM (VAPNIK, 1995) é uma técnica de classificação que tem recebido considerável atenção devido aos resultados promissores observados em muitas aplicações. Uma propriedade interessante do SVM é a sua capacidade de trabalhar bem com dados compostos por muitos atributos, sendo assim menos sensível a problemas com alta dimensionalidade.

O SVM é fundamentado na teoria da aprendizagem estatística, e realiza a classificação no princípio de separação ótima entre duas classes, estabelecendo um limite de decisão usando um subconjunto dos dados de treinamento, conhecido como vetores de suporte. Em outras palavras, o SVM busca um hiperplano aprendido a partir dos dados de treinamento que divide o espaço em duas regiões, uma que comporta os registros pertencentes a uma classe c_a e outra que contém todos os registros da classe c_b (BAEZA-YATES; RIBEIRO-NETO, 2011). O hiperplano aprendido então é utilizado como referência para a classificação de novos registros.

Figura 2.6 – Exemplo de hiperplano no espaço bidimensional.



Fonte: Do autor (2017).

Um exemplo de hiperplano no espaço bidimensional pode ser observado na Figura 2.6. Neste caso particular o hiperplano é uma linha que separa os dados em duas classes. A linha *S* é a que maximiza as distâncias entre registros das duas classes, sendo assim, o hiperplano de decisão. Já as linhas pontilhadas delimitam a região em que a solução é buscada, e por isso são comumente chamadas de hiperplanos delimitadores. Note ainda que o hiperplano *r* também separa os registros em duas classes disjuntas, mas não maximiza a distância entre os vetores suporte.

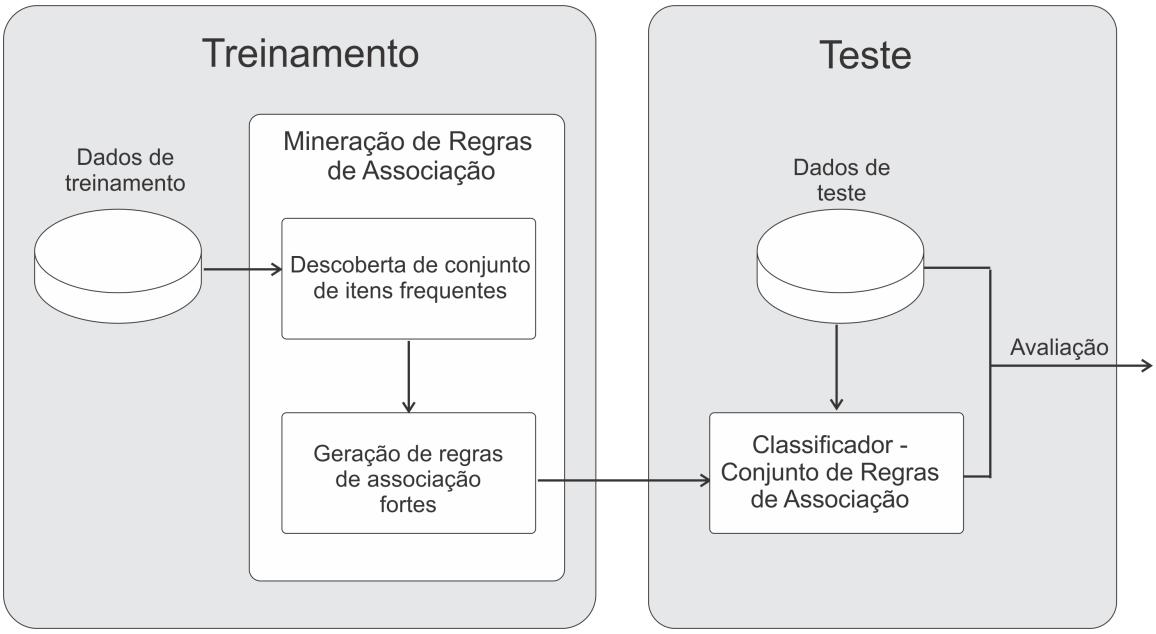
2.4.2 Classificadores associativos

A classificação associativa é uma subárea da mineração de dados que combina as tarefas de previsão e análise associativa com o objetivo de construir modelos de classificação. Essa técnica tem se mostrado uma abordagem promissora, com várias propostas de algoritmos de predição de alta acurácia (WEDYAN, 2014). Segundo Thabtah (2007), as tarefas de classifi-

cação e de análise associativa são similares na mineração de dados, mas enquanto o objetivo da classificação é realizar a predição de classes rotuladas, o objetivo da análise associativa é identificar correlações entre registros de um conjunto de dados.

Essencialmente, uma abordagem de classificação associativa consiste na descoberta de regras de associação a partir dos dados de treinamento. Segundo Tan, Steinbach e Kumar (2009), é comum os algoritmos de mineração de regras de associação serem decompostos em duas fases. Na primeira delas, ocorre a geração de conjuntos de itens frequentes. Na segunda fase, são geradas regras fortes a partir dos itens frequentes da fase anterior. As regras geradas constituem o modelo de predição, que então pode ser utilizado na classificação de novos registros - fase de teste (BOUZOUITA; ELLOUMI, 2011). As fases da classificação associativa podem ser observadas na Figura 2.7.

Figura 2.7 – Classificação Associativa - Adaptado de Thabtah (2007).



Fonte: Do autor (2017).

Várias abordagens de classificação associativa foram propostas nas últimas décadas (LI; HAN; PEI, 2001; ANTONIE; ZAÏANE, 2002; YIN; HAN, 2003; WANG et al., 2011). Oliveira (2014), Oliveira e Pereira (2017) apresentam um método de classificação que utiliza regras de associação para a classificação de ofertas de produtos de lojas *online*.

Regras de associação: é uma técnica proposta em 1993 (AGRAWAL; IMIELINSKI; SWAMI, 1993) muito utilizada em mineração de dados. O objetivo das regras de associação é identificar correlações interessantes nos dados, a partir de padrões frequentes, estruturas de as-

sociação frequentes ou causais. Uma aplicação tradicional das regras de associação é a análise de transações de compras, que busca por padrões de comportamento nas compras de consumidores (BERRY; LINOFF, 1997).

Um exemplo hipotético de uma base de dados de transações de compras é apresentado na Tabela 2.1, em que cada transação é composta por um identificador único e uma lista de produtos adquiridos pelo cliente.

Tabela 2.1 – Exemplo de transações de compra.

TID	Itens
1	{Pão, Café, Cerveja}
2	{Pão, Fraldas, Cerveja, Ovos}
3	{Café, Fraldas, Cerveja, Cola}
4	{Pão, Café, Fraldas, Cerveja}
5	{Pão, Café, Fraldas, Cerveja, Ovos}
6	{Pão, Leite, Fraldas, Cola}

Fonte: Do autor (2017).

A regra {Fraldas $\rightarrow$ cerveja} é um exemplo de regra de associação que pode ser extraída do exemplo da Tabela 2.1 e caracteriza uma forte associação dos dados, indicando que, na maioria das vezes, clientes que compram fraldas também compram cerveja (TAN; STEINBACH; KUMAR, 2009). Além disso, esta regra é facilmente compreendida, demostrando uma característica muito importante da técnica.

Seja $IT = \{it_1, it_2, ..., it_m\}$ um conjunto de m itens distintos observados nas transações e $DT = \{dt_1, dt_2, ..., dt_{|DT|}\}$ uma base de dados constituída por $|DT|$ transações. Considere que cada transação $dt_i \in DT$ é composta por um conjunto de itens, chamado de *itemset*, tal que $it_i \subseteq IT$. Então, uma regra de associação é uma expressão do tipo it_a implica it_b ou Se it_a então it_b, denotada por $it_a \rightarrow it_b$, em que $it_a \cap it_b = \varnothing$.

Contador de suporte: O contador de suporte é uma propriedade importante de um conjunto de itens e define o número de transações em que um conjunto de itens ocorre (AGRAWAL; IMIELINSKI; SWAMI, 1993). Matematicamente o contador de suporte $\sigma(it_x)$ de um conjunto de itens it_x pode ser definido como:

$$\sigma(it_x) = |\{dt_i | it_x \subseteq dt_i, dt_i \in DT\}| \tag{2.6}$$

Para o exemplo da Tabela 2.1, o contador de suporte do conjunto {Pão, Ovos} é igual a dois, pois este conjunto de itens pode ser observado em duas transações. Já o conjunto {Café, Cerveja, Fraldas} ocorre em três transações, e portanto seu contador de suporte é igual a três.

Suporte e Confiança: A força de uma regra de associação pode ser medida em função de seu suporte (Equação 2.7) e sua confiança (Equação 2.8), em que $|DT|$ é o número de transações da base de dados.

$$sup(it_a \rightarrow it_b) = \frac{\sigma(it_a \bigcup it_b)}{|D|} \tag{2.7}$$

$$conf(it_a \rightarrow it_b) = \frac{\sigma(it_a \bigcup it_b)}{\sigma(it_a)} \tag{2.8}$$

Enquanto o suporte determina a frequência em que uma regra é aplicável no conjunto de dados, a confiança indica a frequência em que os itens de it_b ocorrem em transações que contenham it_a. O suporte é uma medida importante, pois regras com suporte baixo podem acontecer simplesmente por coincidência ou não agregar valor para a tomada de uma decisão, sendo muitas vezes utilizado para descartar regras. A confiança por sua vez é importante porque mede a qualidade da inferência realizada, de modo que quanto maior a confiabilidade de uma regra, maior a probabilidade de que um item A esteja em transações que contenham B (TAN; STEINBACH; KUMAR, 2009).

Para as transações da Tabela 2.1 o suporte para a regra {Café, Fraldas} $\rightarrow$ {Cerveja} é $3/6 = 0,5$, enquanto sua confiança é $3/3 = 1,0$.

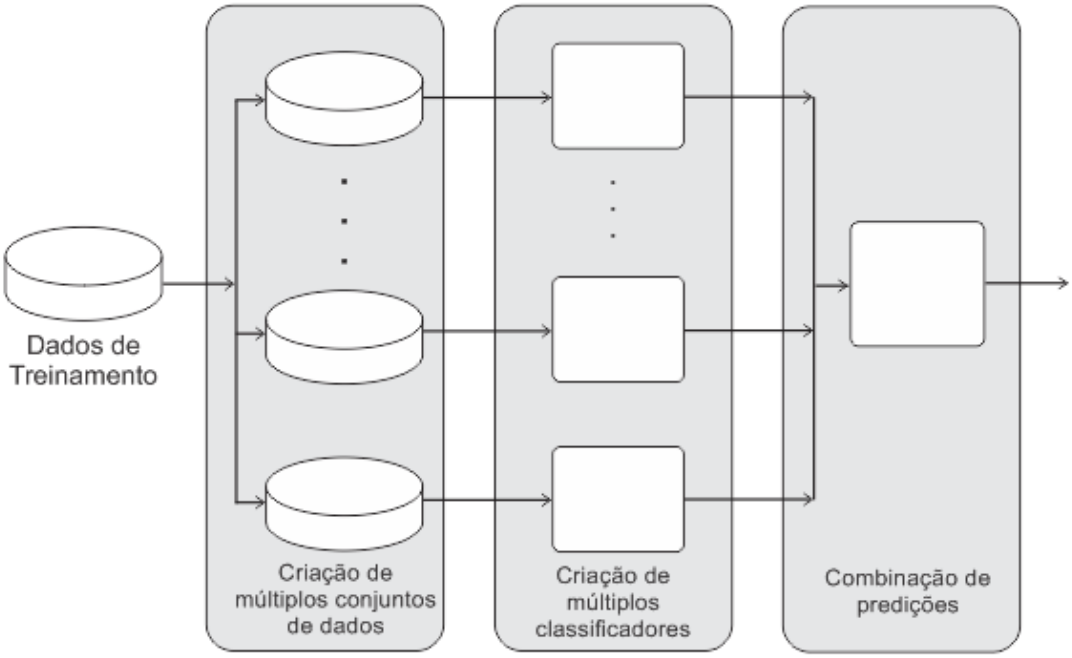
Assim, o problema de mineração de regras de associação pode ser definido como o processo de encontrar todas as regras que tenham suporte $\geq$ *minsup* e confiança $\geq$ *minconf*, onde *minsup* e *minconf* são limites de suporte e confiança, respectivamente, definidos.

2.4.3 *Ensembles*

Um *ensemble* ou combinação de classificadores explora o agrupamento de especialistas com o objetivo de melhorar os resultados da classificação (ROKACH, 2010). Essencialmente, a ideia consiste em combinar a predição de um conjunto de classificadores básicos e independentes em uma decisão final (TAN; STEINBACH; KUMAR, 2009), assim como os humanos tendem a verificar a opinião de vários especialistas antes da tomada de uma decisão (ROKACH,

2010). Uma visão lógica da ideia geral dos ensembles pode ser observada na Figura 2.8, e consiste em criar múltiplos classificadores a partir dos dados originais, e então combinar a previsão de cada um desses classificadores para definir as categorias de novos registros.

Figura 2.8 – Visão lógica dos *ensembles* - Adaptado de Tan, Steinbach e Kumar (2009).



Fonte: Do autor (2017).

Segundo Tan, Steinbach e Kumar (2009), são necessários dois requisitos para que o *ensemble* apresente resultados superiores aos de um único classificador: os classificadores devem ser independentes entre si e os classificadores básicos devem ser melhores que um classificador que realiza predições aleatórias.

Outra característica necessária para que o *ensemble* apresente resultados melhores do que um único classificador é o fato de que cada classificador que constitui o *ensemble* deve cometer erros em registros diferentes. Isto é, se cada classificador comete um erro diferente, então a combinação de vários classificadores pode reduzir o erro global (POLIKAR, 2006). A ideia de conseguir classificadores que façam predições o mais diferente possível é chamada de diversidade, e pode ser obtida das seguintes maneiras:

a) **manipulando o conjunto de treinamento:** usa diferentes conjuntos de treinamento para a geração dos modelos de classificação que constituem o *ensemble*, como é o caso do *bagging* e do *boosting*;

b) **usando diferentes parâmetros:** utiliza diferentes configurações de parâmetros para gerar classificadores diferentes. Um bom exemplo é o observado nas redes neurais artificiais *perceptron* multicamadas, que podem ser treinadas com diferentes pesos de iniciação, número de camadas, etc.;

c) **utilizando técnicas de aprendizagem distintas**: os classificadores que constituem o *ensemble* utilizam diferentes estratégias de aprendizagem. Por exemplo, um *ensemble* poderia ser composto por árvores de decisão, classificadores associativos, classificadores bayesianos, etc.;

d) **utilizando diferente atributos:** cada classificador do *ensemble* é treinado com um subconjunto de características dos dados de entrada.

Uma grande variedade de *ensemble*s é encontrada na literatura, sendo as principais diferenças entre eles a escolha dos classificadores que farão parte do *ensemble* e/ou a maneira de combinação das predições individuais (ROKACH, 2010).

A combinação de predições pode utilizar métodos algébricos, como a média, média ponderada, mediana, soma e soma ponderada. Quando a predição é uma tarefa de classificação, isto é, realiza a predição de valores discretos, é comum se utilizar a votação majoritária e a votação majoritária ponderada (POLIKAR, 2006). Em todo cenário, o objetivo é maximizar o número de registros classificados corretamente e minimizar as classificações incorretas.

A geração dos classificadores também pode ser realizada de diversas maneiras, sendo as estratégias *bagging*, *boosting* e *stacking* e suas respectivas derivações as mais populares. Essas estratégias são brevemente descritas a seguir.

2.4.3.1 *Bagging*

Bagging (BREIMAN, 1996) é a estratégia para a composição de um *ensemble* mais intuitiva e fácil de implementar, e também apresenta boa performance. É particularmente interessante quando poucos dados estão disponíveis para o treinamento. A diversidade nesta estratégia é obtida através de réplicas dos dados de treinamento, isto é, vários subconjuntos são gerados aleatoriamente a partir dos dados de treinamento. Cada um dos subconjuntos possui o mesmo número de registros do conjunto original, e como a amostragem é feita com substituição, alguns registros podem se repetir enquanto outros não são selecionados (TAN; STEINBACH; KUMAR, 2009). Então cada um desses subconjuntos é utilizado para treinar um classificador

diferente, mas que utiliza a mesma estratégia de aprendizagem. O método mais conhecido e largamente utilizado é o *Randon Forest* (BREIMAN, 2001), que é formado por várias árvores de decisões geradas a partir da seleção aleatória de atributos dos dados de entrada.

2.4.3.2 *Boosting*

Diferentemente do *bagging*, a abordagem *boosting* cria uma sequência de classificadores que se diferem entre si pela atribuição de pesos para os registros dos dados de treinamento. Se um registro é classificado incorretamente por um classificador anterior, então ele tem seu peso aumentado, enquanto os registros classificados corretamente têm seus pesos decrementados (ROKACH, 2010). Isso faz com que o foco do próximo classificador da sequência se volte para os registros difíceis de classificar (TAN; STEINBACH; KUMAR, 2009).

Inicialmente, cada registro dos dados de treinamento recebe o mesmo peso e um primeiro classificador é criado, e após a atualização dos pesos dos registros, novos classificadores são criados em um processo iterativo. O modelo final é um algoritmo que combina linearmente os vários classificadores individuais (FREUND; SCHAPIRE, 1996). Uma importante variação *boostig* é conhecida como *Adaboost*.

2.4.3.3 *Stacking*

A abordagem *stacking* usa um classificador de alto nível (nível-2) para combinar os resultados de classificadores de baixo nível (nível-1) (WOLPERT, 1992). Nessa abordagem não é comum que os classificadores utilizem a mesma estratégia de aprendizagem, como é o caso do *bagging* e do *boosting*.

A primeira fase do *stacking* consiste na coleta das predições realizadas para cada registro dos dados de treinamento por cada classificador de nível-1, predições que são transformadas em um novo conjunto de dados junto com suas verdadeiras classes. Esse novo conjunto de dados é tratado como um novo problema de classificação, e então um algoritmo de aprendizagem é utilizado para sua solução.

Os *ensembles* têm sido usados extensivamente na solução de vários problemas, desde os mais simples até aqueles que requerem aprendizagem incremental, fusão de dados, seleção de características e classificação em ambientes dinâmicos (ROKACH, 2010). De qualquer maneira, o objetivo é combinar vários classificadores em busca de resultados mais efetivos, sendo

que a melhor configuração para o *ensemble* deve ser identificada experimentalmente para cada problema a ser solucionado.

2.5 Aprendizagem incremental

Como já discutido, a aprendizagem de máquina tem o objetivo de estudar algoritmos que melhoram seu desempenho para uma determinada tarefa a partir de sua experiência (MITCHELL, 1997). Diversas técnicas de aprendizagem de máquina foram propostas na literatura, sendo aplicadas na resolução de uma grande diversidade de problemas. Entretanto, estas técnicas foram propostas para trabalhar em cenários *batch*, isto é, em cenários nos quais os dados de treinamento estão completamente disponíveis para a geração do modelo de predição, modelo que não é alterado com novos registros de dados. (SPINOSA; CARVALHO; GAMA, 2009).

Entretanto, em muitas aplicações do mundo real, novos registros de dados são gerados o tempo todo. Nesse caso, a aprendizagem deve ocorrer continuamente, como acontece na aprendizagem de humanos, que tem a habilidade de adquirir conhecimento incrementalmente à medida que novos fatos são observados. A aprendizagem de máquina incremental propõe algoritmos de aprendizagem que são atualizados sempre que novos registros de dados se tornam disponíveis (FARIA; CARVALHO; GAMA, 2016).

Seja uma sequência de registros de dados $S = \{(F_1, y_j), ..., (F_N, y_j), (F_{N+1}, y_j), ..., \}$, em que $F_i = \{f_1, f_2, ..., f_n\}$ é um registro de dados composto por n atributos contínuos ou discretos e y_j um rótulo discreto que indica a classe de um registro. Se os registros de dados chegam nos tempos $t_1, t_2, ..., t_n, t_{n+1}, ...$, então a tarefa de classificação incremental pode ser definida como o processo de se aprender uma função alvo que mapeie cada F_i a uma classe y_j, de forma que a predição de novos registros de dados possa ser realizada com alta acurácia. Para isso, algoritmos de classificação tradicionais como o SVM, as árvores de decisões, classificadores associativos precisam ser adaptados.

Duas abordagens são tradicionalmente utilizadas para a implementação da aprendizagem incremental. A primeira delas utiliza um único modelo de predição, que é atualizado dinamicamente para agregar os novos dados. Na segunda abordagem, são utilizados *ensemble* de classificadores. Nessa abordagem, novos classificadores são criados a partir dos novos dados e são adicionados ao *ensemble* (MASUD et al., 2013).

Geralmente os algoritmos incrementais apresentam duas fases bem definidas (PAIVA, 2014). Na primeira delas, denominada fase *offline*, é gerado um modelo de decisão com base em

um conjunto de dados de treinamento. Esse modelo estabelece uma função que mapeia as informações obtidas no conjunto de treinamento (conceitos) em possíveis saídas para o algoritmo (MITCHELL, 1997), assim como é feito nas técnicas de aprendizagem de máquina tradicionais.

Na segunda fase, denominada fase *online*, acontece a predição de um novo registro de dados. Os registros que podem ser associados com os conceitos observados na fase *offline* são classificados. Já os registros que não puderam ser classificados são ditos desconhecidos, uma vez que o modelo de decisão não possui informação suficiente para classificá-los. Um registro desconhecido pode ser utilizado imediatamente para modelar novos conceitos (ALBERTINI; MELLO, 2007) ou pode ser usado em conjunto com outros registros desconhecidos para modelar padrões que definem novos conceitos ou mesmo mudanças em conceitos já conhecidos pelo modelo de aprendizagem *offline* (SPINOSA; CARVALHO; GAMA, 2009).

Uma propriedade desejável para os algoritmos incrementais é a capacidade de adaptar o modelo de treinamento com os novos dados que recebe, já que a evolução de conceitos, a evolução de atributos e a mudança de conceitos são comuns em muitos problemas reais (MASUD et al., 2013). Assim, um algoritmo de aprendizagem incremental deve ser capaz de aprender e atualizar o modelo de predição com os novos dados, ser capaz de criar novas classes quando observar a evolução de conceitos, ser capaz de se adaptar para representar a mudanças que ocorreram nos conceitos e lidar com o surgimento de novos atributos que ainda não foram observados pelo modelo de decisão.

É importante ressaltar que independente da utilização de um único modelo ou da utilização de *ensembles* para a implementação da aprendizagem incremental, o tratamento de mudanças no conceito e evolução no conceito depende do algoritmo utilizado na aprendizagem. Por exemplo, Ferreira et al. (2014) utilizam um classificador associativo e a detecção de novos conceitos é feita com evidências obtidas no conjunto de regras de associação do modelo de treinamento. Já em Masud et al. (2013), é utilizado um *ensemble* de classificadores baseado no KNN, e a detecção de mudanças no conceito e evolução do conceito é realizada com referência aos centróides das classes.

O tratamento do *feature evolution*, por outro lado, pode ser realizado de três maneiras (MASUD et al., 2013). Uma delas consiste em manter o número de atributos fixo. Assim, os atributos dos novos registros de dados devem ser mapeados para o conjunto de atributos observados na fase *offline* do algoritmo. A segunda alternativa geralmente é aplicável quando se implementa a aprendizagem incremental com o uso de *ensemble* de classificadores. Nesse caso,

cada um dos classificadores que compõe o *ensemble* possui seu próprio conjunto de atributos, mas ainda assim é necessário o mapeamento dos atributos de um novo registro para os atributos do modelo escolhido para realizar a classificação. A terceira opção consiste na expansão do conjunto de atributos tanto do modelo de predição quanto do novo registro de dados. Nesse caso, o novo conjunto de atributos que descreve os registros de dados passa a ser a união do conjunto de atributos já observados pelo modelo de predição e do conjunto de atributos do novo registro.

Note que todas as considerações feitas sobre a aprendizagem supervisionada incremental são facilmente adaptadas para os métodos de *clustering*, que utilizam aprendizagem não supervisionada, ou mesmo para modelos de aprendizagem híbridos.

2.6 Trabalhos relacionados

Nesta seção, são discutidos os trabalhos propostos por Pereira, Silva e Esmin (2014) e Oliveira (2014) e Oliveira e Pereira (2017). Este último é o algoritmo base adaptado para permitir a sua atualização incremental e assim ser usado no método para a resolução de entidades apresentado neste trabalho. Também são apresentados trabalhos disponíveis na literatura que tratam a resolução de entidades como um problema que demanda por uma abordagem incremental para sua solução. Por fim são discutidos alguns trabalhos que propõem abordagens incrementais com suporte à detecção de nova classes, mesmo que estes não tenham sido aplicados diretamente à resolução de entidades.

2.6.1 Trabalhos de Pereira, Silva e Esmin (2014), Oliveira (2014) e Oliveira e Pereira (2017)

O método apresentado por Oliveira (2014), Oliveira e Pereira (2017) é um método para classificação de ofertas de produtos em lojas de comércio eletrônico baseado em regras de associação. Esse método é um desdobramento do método proposto em Pereira, Silva e Esmin (2014), que utiliza regras de associação para a desambiguação de títulos de veículos de publicação.

O método apresentado por Pereira, Silva e Esmin (2014) é um método de aprendizagem de máquina supervisionado que usa regras de associação (AGRAWAL; SRIKANT, 1994) aplicado ao caso particular de resolução de entidades que tem o objetivo de desambiguar nomes de veículos de publicações em citações bibliográficas. Na proposta, o modelo gerado na fase de

treinamento é um conjunto de regras de associação na forma $X \rightarrow pv_i$, em que X é um conjunto de palavras chaves e pv_i é o nome de um veículo de publicação. Na fase de teste, para cada registro são gerados conjuntos de palavras-chave a partir do nome do veículo de publicação da citação. Esses conjuntos de palavras-chave são buscados nos antecedentes das regras geradas na fase de treinamento, e os consequentes dessas regras são utilizados em um esquema de votação majoritária para desambiguar o título do veículo de publicação do registro de teste.

O método proposto por Oliveira (2014) e Oliveira e Pereira (2017), por sua vez, propõe algumas alterações ao método de Pereira, Silva e Esmin (2014) e o aplica na classificação de ofertas de produtos, outro caso particular de resolução de entidades. Nesse trabalho, durante a fase de treinamento são utilizadas as descrições textuais de produtos ofertados em lojas de comércio eletrônico que geram regras de associação na forma $X \rightarrow c_i$. X é um conjunto de palavras-chave obtidos da descrição da oferta e c_i é uma classe que identifica unicamente cada produto. Então essas regras são usadas na fase de teste para predizer os produtos de novas ofertas. A predição busca os conjuntos de palavras-chave obtidas das descrições dos registros de teste nos antecedentes das regras geradas na fase de treinamento, e o consequente da regra que obtiver mais votos é definido como o produto da oferta considerada no teste.

Algumas diferenças podem ser observadas entre os dois trabalhos. Primeiramente, em Oliveira (2014) e Oliveira e Pereira (2017) foi introduzido o conceito de suporte na fase de treinamento, com o objetivo de se evitar a geração de regras que podem acontecer simplesmente por coincidência e não agregam valor à tomada de decisão. Outras diferenças entre os métodos podem ser observadas na fase de teste, já que em Oliveira (2014) e Oliveira e Pereira (2017) os conjuntos de *tokens* de tamanho $k+1$ são gerados somente se a classe de um registro de teste não pode ser predita com os conjuntos de *tokens* de tamanho k na iteração anterior. Os métodos também se diferem na estratégia utilizada para a classificação de registros de dados para as quais não foram encontradas regras ou na ocorrência de empate entre duas ou mais classes. Em Oliveira (2014) e Oliveira e Pereira (2017) a predição através de similaridade considera todos os registros de treinamento em sua tomada de decisão, tanto para o caso em que ocorre empate entre duas ou mais regras quanto no caso em que não são encontradas regras. Já em Pereira, Silva e Esmin (2014), se a predição não pode ser realizada com regras devido a um empate, somente os registros das classes que empataram são consideradas na predição por similaridade. Ademais, o método proposto por Oliveira (2014), Oliveira e Pereira (2017) apresenta e avalia algumas alternativas para o método básico, que foram verificadas com o objetivo de melhorar

a complexidade computacional da proposta e abordar diferentes facetas de sua aplicação no problema de classificação de ofertas de produtos.

O principal classificador utilizado no método apresentado neste trabalho é o método proposto por Oliveira (2014) e Oliveira e Pereira (2017), que foi adaptado para permitir sua atualização incremental. A atualização do classificador associativo consiste na agregação de novas regras de associação obtidas de novos registros de dados ao modelo gerado na fase de treinamento. O auto treinamento do classificador associativo com novos registros de dados também pode levar à eliminação de regras, caso a regra deixe de possuir 100% de confiança ou deixe de atender ao suporte mínimo. O auto treinamento do classificador associativo é descrito em detalhes no Capítulo 3.

2.6.2 Resolução de entidades

A resolução de entidades é um importante passo para a integração de bases de dados e no processo de limpeza de dados. A dificuldade e importância do problema de resolução de entidades e suas variações têm recebido grande atenção em diversas pesquisas em várias áreas do conhecimento nas últimas décadas. Em muitos casos, o problema pode ser apresentado com outros nomes, como *record linkage* (FELLEGI; SUNTER, 1969), *deduplication* (CARVALHO et al., 2006), *citation matching* (LEE et al., 2007), *identity uncertanty* (PASULA et al., 2002), *merge/purge* (HERNÁNDEZ; STOLFO, 1995), *entity matching* (KÖPCKE; RAHM, 2010) e *product matching* (KÖPCKE et al., 2012).

Eficiência e qualidade dos resultados são desafios tradicionais na resolução de entidades. As pesquisas relacionadas à eficiência, cujo foco é a redução do tempo de execução de algoritmos para a resolução de entidades, exploram técnicas de *blocking* (BALAJI et al., 2016; PAPADAKIS; PAPASTEFANATOS; KOUTRIKA, 2014; MCCALLUM; NIGAM; UNGAR, 2000; WINKLER, 2005; BAXTER; CHRISTEN, 2003) e/ou frameworks de programação paralela como o MapReduce para atingir seus objetivos (EFTHYMIOU et al., 2017; MALHOTRA; AGARWAL; SHROFF, 2014a; SARMA et al., 2012). Por outro lado, muitas pesquisas se preocupam com a obtenção de melhores resultados, e para isso exploram informações adicionais. Tais informações podem explorar contextos específicos (BHATTACHARYA; GETOOR, 2004), relações entre entidades (BHATTACHARYA; GETOOR, 2007b; KALASHNIKOV; MEHROTRA, 2006), comportamento das entidades (YAKOUT et al., 2010), inteligência humana na forma de *crowdsourcing* (VESDAPUNT; BELLARE; DALVI, 2014; WHANG;

LOFGREN; GARCIA-MOLINA, 2013; WANG et al., 2013) ou mesmo bases de conhecimentos externos, como ontologias e máquinas de busca Web (WELCH; SANE; DROME, 2012; NURAY-TURAN; KALASHNIKOV; MEHROTRA, 2012; PEREIRA et al., 2011; ELMACIOGLU et al., 2007).

O surgimento de aplicações Web atuais tem motivado a pesquisa de novas abordagens para a resolução de entidades, como é o caso de métodos incrementais para a resolução de entidades, da resolução de entidades progressiva e da resolução de entidades *real-time*.

A abordagem proposta neste trabalho é um método incremental para a resolução de entidades. Abordagens incrementais têm o objetivo de manter a resolução de entidades atualizada sempre que novos dados estão disponíveis. Abordagens incrementais foram propostas por Gruenheid, Dong e Srivastava (2014), Whang e Garcia-Molina (2014), Welch, Sane e Drome (2012), Costa, Manco e Ortale (2010) e Bilenko, Basil e Sahami (2005), são os mais próximos ao método proposto neste trabalho.

Em Bilenko, Basil e Sahami (2005), os autores consideram um caso específico do problema, o *match* de produtos em sistemas de comparação em lojas online. O método proposto requer o aprendizado de uma função de similaridade entre pares de registros em *data streams*. Diferente da abordagem proposta neste trabalho que trata os registros de dados com um único atributo, o método desses autores consiste em estabelecer uma combinação linear de funções de similaridade básicas aplicadas a dados estruturados. Além disso, os autores não avaliaram a detecção de novas entidades. Somente citam que um novo registro pode ser agregado aos *clusters* existentes caso a semelhança entre o registro a ser predito e os registros de algum *cluster* seja superior a um limiar previamente estabelecido. Caso contrário, o registro é adicionado a um novo *cluster*.

Costa, Manco e Ortale (2010) também tratam a resolução de entidades como um problema de *clustering*, de forma que dado um conjunto de registros, o algoritmo os particiona em grupos de referências à mesma entidade. O mapeamento do novo registro em um *cluster* transforma o registro em um conjunto de chaves indexadas, e em seguida o registro é atribuído ao *cluster* apropriado através de classificação, isto é, o novo registro é atribuído ao *cluster* cujos registros são mais similares. A utilização da ideia de vizinhos mais próximos ao registro permite que tal método identifique a qual *cluster* o registro deve ser associado eficientemente, já que nesse caso não é necessário considerar toda a base de dados.

Em Welch, Sane e Drome (2012), os autores projetaram um sistema de resolução de entidades aplicável em sites Web. O sistema usa uma máquina de busca proprietária para selecionar registros que se casam e assim derivam uma decisão de deduplicação. As operações incrementais nesse sistema permitem que um registro de dados seja adicionado a um *cluster* já existente, ou formar um novo *cluster*. A eficiência do método é alcançada através de técnicas de *blocking*.

O trabalho apresentado por Gruenheid, Dong e Srivastava (2014) propõe uma abordagem para a resolução incremental de entidade que permite o agrupamento de registros durante a atualização de *clusters* existentes. Adicionalmente, a proposta usa novas evidências dos novos dados para corrigir possíveis erros da solução. Sua solução usa um grafo de similaridades entre os registros e realiza *clustering* incremental nesse grafo. Os algoritmos dos autores, instanciados em dois métodos de *cluster*, agrupam subconjuntos dos dados em vez de usar todos os registros, usando uma abordagem gulosa que mescla e divide *clusters* conectados aos registros atualizados e movimenta registro entre *clusters*.

O foco do trabalho proposto em Whang e Garcia-Molina (2014) foi a avaliação da resolução de entidades no caso em que regras de casamento evoluem com o passar do tempo, isto é, a lógica que examina e compara registros muda. Os autores brevemente discutem a evolução dos dados. No método proposto neste trabalho a evolução de regras não é abordada, e nem são utilizados métodos de *clustering*.

O objetivo do método proposto por Malhotra, Agarwal e Shroff (2014b) é agrupar documentos em entidades de modo que todos os documentos do mesmo *cluster* correspondam a uma mesma entidade do mundo real. Na fase de *bloking*, os autores usam similaridade textual e referências entre documentos para agrupar documentos que são prováveis referências à mesma entidade, e assim todos os pares de documentos não precisam ser verificados. A partir dos *clusters* gerados, as referências em cada *cluster* são resolvidas e finalmente um algoritmo é utilizado para encontrar os componentes conexos no grafo que representa os documentos, com o objetivo de agrupar referências a entidades que foram associadas a mais de um *cluster*. Os autores também mostram que um novo conjunto de documentos pode ser agregado à solução inicial incrementalmente, sem ter que refazer todo o processamento usando tanto os dados antigos quanto os novos, e sem grande perda de qualidade.

Todos os trabalhos descritos nos parágrafos anteriores usam a abordagem tradicional para resolução de entidades, onde são comparados todos os pares de registros pertencentes a um

bloco. Nessas comparações, funções de similaridade distintas são usadas em cada atributo do registro estruturado e então um método de *clustering* é utilizado para decidir a qual entidade os registros do par correspondem. No método proposto neste trabalho, é utilizada uma abordagem diferente. Um *ensemble* de classificadores multiclasses é treinado para identificar um conjunto inicial de entidades e, incrementalmente, com a disponibilidade de novos dados, o método identifica novas entidades e atualiza a solução com os melhores registros.

Outras propostas para a resolução de entidades já utilizaram a ideia de *ensembles*. Balaji et al. (2016) usam *ensemble* para combinar técnicas de *blocking* e assim gerar uma nova estratégia, menos impactada por variações dos dados. Papadakis, Papastefanatos e Koutrika (2014) também utilizam *ensemble* na proposta de uma abordagem de *bloking*. Em Yi et al. (2017), os autores tratam a resolução de entidades como um problema de classificação binária, que classifica os registro em um par como similares ou não. A classificação é dada não por um classificador, mas por um conjunto deles, cada um gerado por um conjunto seleto de *features*. Neste trabalho, o *ensemble* de classificadores foi utilizado primariamente na detecção de referências a entidades previamente desconhecidas pelo algoritmo e finalmente para predizer a qual entidade um registro faz referência naqueles casos em que o classificador prioritário não pode tomar uma decisão.

A resolução de entidades progressiva tem como objetivo identificar e resolver o máximo de pares de registros que fazem referência a uma mesma entidade em uma base de dados, enquanto tenta não resolver pares de registros que não fazem referência (ALTOWIM; KALASHNIKOV; MEHROTRA, 2014). Para isso, uma abordagem progressiva prioriza quais partes dos dados resolver a fim de maximizar o número de entidades resolvidas. Efthymiou, Stefanidis e Christophides (2016), Papenbrock, Heise e Naumann (2015), Whang, Marmaros e Garcia-Molina (2013) apresentam métodos para a resolução de entidades progressiva.

Na resolução de entidades *real-time*, o objetivo é realizar o *matching* de um registro com registros da mesma entidade em uma base de dados no menor tempo possível (RAMADAN et al., 2015; BANDA, 2016). Dey, Mookerjee e Liu (2011) e Bhattacharya, Getoor e Licamele (2006) são alguns autores que também apresentam propostas para a resolução de entidades real-time.

Adicionalmente a essas abordagens, a comunidade que estuda soluções para a resolução de entidades tem explorado novas direções. Em Dallachiesa et al. (2013), os autores propõem um sistema genérico para a limpeza e reparação de dados, sistema com abstrações de progra-

mação que permitem aos usuários especificar os tratamentos a serem realizados nos dados. Já Stonebraker et al. (2013) apresentam um sistema para recuperação de dados que usa algoritmos de aprendizagem de máquina com entradas fornecidas por humanos para realizar integração de dados e resolução de entidades.

2.6.3 Classificação com detecção de novas classes

Outra área de estudo diretamente relacionada à proposta deste trabalho é a classificação de *data streams*, que inclusive se preocupa com a proposta de classificadores com detecção de novas classes. Masud et al. (2013) e Faria, Carvalho e Gama (2016) são trabalhos que propõem classificadores com a habilidade de detectar novas classes. No entanto, essas abordagens são capazes de identificar pequenos números de novas classes que contêm muitos registros, e não foram avaliados em dados não estruturados. O objetivo desses trabalhos é tratar os problemas *concept-drift*, *concept-evolution* e *feature-evolution*. O método proposto neste trabalho naturalmente trata tais fenômenos através do mecanismo de auto treinamento e detecção de novas classes.

MÉTODO PROPOSTO

Neste capítulo, é apresentado o método incremental para resolução de entidades proposto. A apresentação se inicia com a formulação do problema e é seguida pela descrição do método. Por fim é discutida a complexidade computacional do método.

3.1 Formulação do problema

Dado um conjunto de dados $I = \{d_1, d_2, ..., d_n\}$, o problema de resolução de entidades consiste em (i) determinar o conjunto $\mathscr{E} = \{e_1, e_2, ..., e_p\}$ de entidades do mundo real relacionadas ao conjunto de registros I e (ii) associar a cada registro d_i a entidade $e_k \in \mathscr{E}$ correspondente (correta). Note que o número de entidades distintas p não é fornecido como entrada. No método proposto neste trabalho, a resolução de entidades é tratada como um problema de classificação multiclasses. Para isso, cada entidade e_k é modelada como uma classe c_k, nos algoritmos de aprendizagem supervisionada.

O conjunto de dados I é subdivido em dois subconjuntos disjuntos $\mathscr{D}$ e $\mathscr{T}$. $\mathscr{D}$, denominado conjunto de treinamento, é composto por registros para os quais a classe correspondente é conhecida. O conjunto de registros de treinamento se relaciona com o conjunto inicial $C = \{c_1, c_2, ..., c_l\}$ de l classes, $l < p$, e é utilizado para produzir um modelo de aprendizagem, relacionando cada registro à classe correta. Essa é a fase de treinamento do algoritmo de classificação.

O subconjunto de dados $\mathscr{T}$, denominado conjunto de teste no problema de classificação, é um conjunto de registros para os quais a classe não é conhecida. Na fase de teste do algoritmo de classificação, o modelo de aprendizagem é usado para realizar a predição da classe à qual um novo registro $d_i \in \mathscr{T}$ faz referência.

Métodos incrementais para a resolução de entidades têm o objetivo de manter a resolução de entidades atualizada sempre que novos dados estão disponíveis. Em outras palavras, o conjunto de dados $\mathscr{T}$ deve ser utilizado para atualizar a solução, de forma que não seja necessário realizar toda a resolução de entidades novamente considerando tanto dados antigos quantos os novos, apenas atualizar uma solução já existente com as informações dos novos dados (RAMADAN et al., 2015).

Seja $\Delta\mathscr{T}$ um conjunto de operações *insert* realizadas em algum momento do tempo, caracterizado pela adição de novos registros d_i ao conjunto de dados de treinamento $\mathscr{D}$. Esse conjunto de operações é denominado **incremento**, e a aplicação do incremento $\Delta\mathscr{T}$ a $\mathscr{D}$ é

denotado por $\Delta\mathscr{T} + \mathscr{D}$. Segundo Gruenheid, Dong e Srivastava (2014), as operações em um incremento também podem ser do tipo *delete*, caracterizada pela remoção de um registro d_i de $\mathscr{D}$ e do tipo *change*, caracterizada pela modificação de um ou mais atributos de um registro $d_i \in \mathscr{D}$. No entanto, neste trabalho tratou-se somente operações do tipo *insert*, e nesse caso $|\mathscr{D} + \Delta\mathscr{T}| = |\mathscr{D}| + |\Delta\mathscr{T}|$. No método proposto, a adição dos novos registros à resolução de entidades inicial é feita com o auto treinamento do modelo de aprendizagem, conforme discutido na Seção 3.5.

Adicionalmente, no método proposto é apresentado um mecanismo para o tratamento da evolução do conceito. Após a fase de treinamento, o conjunto inicial $C = \{c_1, c_2, ..., c_l\}$ possui l classes, com as quais os registros do conjunto de treinamento se relacionam. Entretanto, alguns dos registros do incremento $\Delta\mathscr{T}$ podem não fazer referência a nenhuma dessas classes iniciais, de forma que uma ou mais novas classes precisem ser criadas, e assim $C = \{c_1, c_2, ..., c_l, c_{l_1}, c_{l+2}, ...\}$. Idealmente, ao fim da fase de teste, isto é, ao fim da aplicação do incremento $\Delta\mathscr{T}$, o conjunto C de classes será o próprio conjunto de entidades $\mathscr{E}$. Quando novas classes são detectadas pelo algoritmo, o mecanismo de auto treinamento agrega essas novas classes ao modelo de aprendizagem inicial, através da adição dos novos registros de dados ao conjunto de dados inicial.

Embora este trabalho não tenha se preocupado com o tratamento da evolução de atributos, o modelo de aprendizagem utilizado trata naturalmente esse fenômeno. Também não foi foco deste trabalho o tratamento da mudança de conceito, contudo o auto treinamento do modelo de aprendizagem utilizado é capaz de tratar alguns tipos de ocorrência desse fenômeno, especificamente aqueles conhecidos como mudança de conceitos virtuais, nos quais as definições das classes sofrem alterações (GAMA et al., 2014). Contudo, para finalizar o tratamento da mudança de conceito, é necessário considerar os outros tipos de operações em um incremento, isto é, o algoritmo deve ser capaz de se atualizar na presença de operações do tipo *delete* e *change*, descritas anteriormente.

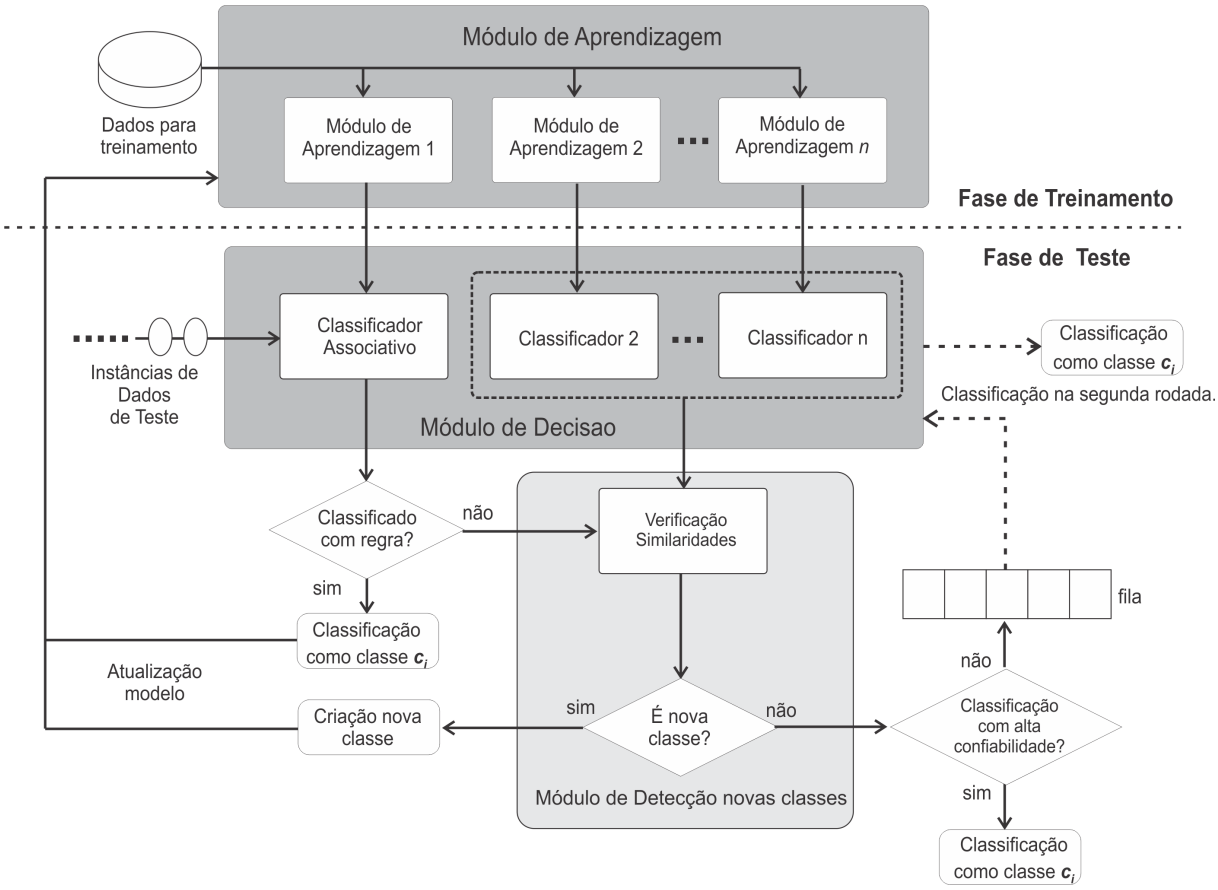
O método proposto neste trabalho é denominado AssocIER, e é detalhado nas próximas seções.

3.2 Visão geral

O método AssocIER é ilustrado na Figura 3.1, e é baseado em um *ensemble* de classificadores. Seu principal classificador é o classificador associativo proposto por Oliveira (2014) e Oliveira e Pereira (2017), modificado para permitir sua atualização incremental. Esse classifica-

dor é o único fixo no *ensemble* proposto, e os demais algoritmos de classificação, denominados classificadores auxiliares, são configuráveis. Qualquer classificador *lazy* ou incremental pode ser usado como classificador auxiliar. O classificador associativo em conjunto com os classificadores auxiliares compõem o Módulo de Aprendizagem, e os modelos gerados por esses algoritmos compõem os Módulo de Decisão do método AssocIER.

Figura 3.1 – Método incremental para a resolução de entidades.



Fonte: Do autor (2017).

Na fase de treinamento, cada classificador no *ensemble* é treinado com o conjunto de dados de treinamento $\mathscr{D}$. A fase de teste é dividida em duas rodadas. Na primeira, todos os registros do incremento $\Delta\mathscr{T}$ que podem ser decididos por regras de associação pertencentes ao classificador associativo são classificados e usados na atualização do modelo de aprendizagem. Os registros que não puderem ser classificados com regras são apresentados ao módulo de detecção de novas classes. Esse módulo verifica os classificadores auxiliares do *ensemble*, e a decisão particular de cada um deles é considerada pelo módulo de detecção de novas classes na tomada de uma decisão sobre a classificação do registro d_l testado. Para cada registro, se a classificação indicada pelos classificadores auxiliares é de baixa confiança então o registro

é classificado como pertencente a uma nova classe. Um novo rótulo é criado no modelo de aprendizagem e o registro é usado para atualizar o modelo de aprendizagem. Por outro lado, se a classificação sugerida pelos classificadores auxiliares é de alta confiança, então o registro é classificado como pertencente a uma das classes já conhecidas pelo modelo de aprendizagem, que também é atualizado com esse registro. Caso contrário, a classificação do registro é adiada, isto é, o registro é adicionado a uma fila e será classificado na segunda rodada do teste, depois da verificação de todos os demais registros do incremento $\Delta\mathscr{T}$.

Na segunda rodada do teste, os registros da fila são apresentados primeiramente pelo classificador associativo. Os registros que não podem ser preditos por esse classificador são classificados pelos classificadores auxiliares, pela classe com mais votos ou por um classificador preferencial. Dessa vez, não há a identificação de novas classes e nenhum registro do incremento é utilizado na atualização do modelo de aprendizagem. É esperado que os registros que tiveram sua predição adiada sejam melhor classificados nessa rodada, devido à atualização do modelo de aprendizagem na primeira rodada do teste.

A seguir o método AssocIER é melhor detalhado.

3.3 Fase de treinamento

Na fase de treinamento, os registros do conjunto $\mathscr{D}$ são apresentados como exemplos de treinamento a cada um dos classificadores do *ensemble*, de forma que cada classificador gere seu próprio modelo de aprendizagem. A geração do modelo baseado em classificação associativa é descrita nos próximos parágrafos. . .

Seja d_{j,c_i} um registro do conjunto de treinamento, representado por uma *string* j que descreve o registro e um rótulo de classe c_i com a qual o registro se relaciona. Seja $R_{c_i}^{d_j}$ o conjunto de regras de associação gerado a partir de d_j, e R_{c_i} o conjunto de regras de associação que predizem a classe c_i originalmente formadas por todas os registros dessa classe. O modelo de aprendizagem gerado é composto pelo conjunto de regras R, tal que $R_{c_i}^{d_j} \subseteq R_{c_i} \subseteq R$.

O classificador associativo explora associações entre *tokens* que identificam unicamente os registros em cada classe. As regras de associação são da forma $X \rightarrow c_i$, em que $X \subseteq F$ é um conjunto de atributos e $c_i \in C$ é um rótulo de classe conhecido. No classificador associativo usado neste trabalho, as regras devem apresentar 100% de confiança, isto é, $X \rightarrow c_i$ é uma regra de associação no modelo de aprendizagem se os registros de apenas uma classe c_i possuem os *tokens* do antecedente X. Adicionalmente, o classificador associativo verifica o suporte da

regra, de forma que a regra $X \rightarrow c_i$ tem suporte s em $\mathscr{D}$ se $s\%$ dos registros da classe c_i contém X. O suporte é medido somente entre os registros da mesma classe, diferentemente do suporte calculado a partir de todos os registros do conjunto, como proposto em Agrawal e Srikant (1994).

O Algoritmo 1 descreve os passos de treinamento do classificador associativo. Sua entrada é um conjunto para treinamento $\mathscr{D}$ e um suporte mínimo *ms*, e sua saída é o conjunto R de regras de associação geradas. Para a geração das regras de associação foi utilizada uma estrutura de índice invertido (BAEZA-YATES; RIBEIRO-NETO, 2011), no qual o vocabulário é um conjunto de *features* F e a lista de ocorrências de cada *feature* f_i contém as classes em C em que f_i ocorre. Esse índice invertido é criado nas linhas 1-6 do Algoritmo 1, e sua utilização faz o processo de geração das regras de associação mais eficiente, já que somente regras com 100% de confiança precisam ser geradas. Note que dessa forma todos os conjuntos de itens não precisam ser criados antes de gerar as regras de associação, como é feito por exemplo no Algoritmo Apriori (AGRAWAL; SRIKANT, 1994).

Então, para a geração das regras de associação (Lines 7-23) é suficiente que seja verificada a lista de ocorrências do conjunto de itens no índice invertido. Se a lista de ocorrências de um conjunto de itens é maior que 1, esse conjunto de itens não gera uma regra com 100% de confiança. Para conjuntos de itens com mais de uma *feature*, é suficiente realizar a interseção das listas de ocorrência de cada *feature*. Um conjunto de itens que contém k *features* é chamado k-itemset, e a função *GenItemSets* gera os k-itemsets combinando os $(k-1)$-itemsets da iteração anterior. A função *RemoveItemSet* implementa uma estratégia de poda que evita a combinação de um k-itemset na próxima iteração do algoritmo, pois se um k-itemset gera uma regra de associação com 100% de confiança, então qualquer l-itemset que contenha esse k-itemset, $l > k$, também gera. Adicionalmente, por razões de eficiência, o número de iterações, m (Linha 11) foi limitado a no máximo 3. Esse valor provou-se suficiente para obter resultados de qualidade, como demonstrado em Oliveira (2014) e Oliveira e Pereira (2017).

.

Exemplo 1. Considere os 4 registros de treinamento d_{j,c_i} da Figura 3.2 e suas respectivas classes e um suporte mínimo $ms = 0.0$. Após a execução do Algoritmo 1, serão geradas as regras de associação r_1 a r_4 da figura. Se o suporte mínimo fosse maior que 50%, a regra r_3 não seria gerada.

Algoritmo 1: Fase de treinamento.

Require: Examples for training D
Require: Minimum support ms
Ensure: A set of rules R

```
1:  for each record d_{j,c_i} ∈ D do
2:      S_0 ← Tokenize(d_{j,c_i})
3:      for each token t_k ∈ S_0 do
4:          InsertInvertedIndex(t_k, j, c_i)
5:      end for
6:  end for
7:  R ← ∅
8:  for each record d_{j,c_i} ∈ D do
9:      S_0 ← Tokenize(d_{j,c_i})
10:     m ← Length(S_0)
11:     for (k ← 1; k ≤ m; k++) do
12:         S_k ← GenItemSets(k, S_{k-1})
13:         for each k-itemset it ⊆ S_k do
14:             if SizeOccurrenceList(it) = 1 then
15:                 //100% confidence rule
16:                 if SupportRule(it → c_i) ≥ ms then
17:                     InsertRule(it → c_i, R)
18:                 end if
19:                 RemoveItemSet(it, S_k)
20:             end if
21:         end for
22:     end for
23: end for
24: return R
```

Ainda na fase de treinamento, o mesmo conjunto de dados $\mathscr{D}$ também é usado para treinamento dos classificadores auxiliares do *ensemble*. Nos resultados apresentados neste trabalho foram usados um classificador baseado na similaridade do Cosseno (SALTON; MCGILL, 1983) e outro baseado na similaridade de Jaccard (JACCARD, 1901). Na fase de treinamento, são criadas as estruturas necessárias para esses classificadores. No caso do Cosseno, os pesos dos *tokens* são computados usando a combinação TF-IDF. Por razões de eficiência, o classificador baseado em Cosseno foi implementado usando a função de ranqueamento proposta em Persin (1994).

3.4 Fase de teste

Na fase de teste, cada registro de teste do incremento $\Delta\mathscr{T}$ é primeiramente apresentado ao classificador associativo, e nos casos em que esse classificador não é capaz de tomar uma

Figura 3.2 – Exemplo de execução do método proposto.

- - - - *Conjunto de treinamento.* - - - -
d_{1,c_1}: iPhone 7
d_{2,c_1}: Apple iPhone 7
d_{3,c_2}: Samsung Galaxy S7 Edge
d_{4,c_3}: Samsung Galaxy S7
- - - - - *Regras de associação* - - - - -
$r_1 : \{iPhone\} \rightarrow c_1$
$r_2 : \{7\} \rightarrow c_1$
$r_3 : \{Apple\} \rightarrow c_1$
$r_4 : \{Edge\} \rightarrow c_2$

- - - - - - *Conjunto de teste* - - - - - - - - - - - -	*Predição* - - - - - -
d_5: S7 Edge	c_2
d_6: Moto G4	c_4 (nova classe)
d_7: Smartphone Moto G4	c_4
d_8: Samsung Galaxy S7	c_3

- - - - *Regras de associação adicionadas na fase de teste* - - - -
$r_5 : \{Moto\} \rightarrow c_4$
$r_6 : \{G4\} \rightarrow c_4$
$r_7 : \{Smartphone\} \rightarrow c_4$

Fonte: Do autor (2017).

decisão, o registro é apresentado aos classificadores auxiliares. A fase de teste acontece em duas rodadas e ambas são descritas a seguir. Para a classificação de *data streams*, os dados podem ser divididos em blocos para formar o conjunto de teste, de acordo com um número de registros ou um limite de tempo.

3.4.1 Fase de teste - Primeira rodada

O Algoritmo 2 descreve a primeira rodada do fase de teste. Essa fase recebe como entrada um registro de dados d_l, o conjunto de regras de associação R e uma lista $\mathscr{C}$ de classificadores auxiliares que compõem o *ensemble*. Seu retorno é a predição do registro d_l ou d_l adicionada à fila Q, caso em que a predição do registro ocorrerá na segunda rodada do teste, após a predição dos demais registros de teste.

Nesta rodada, o registro primeiramente é apresentado ao classificador associativo. Cada k-itemset gerado pela função *GenItemSet* na 5 linha do Algoritmo 2 a partir da descrição de d_l é comparado com os antecedentes das regras de associação em R, gerando o conjunto de regras candidatas R_d (Linhas 1-10). Um esquema de votação entre as classes verificadas nos conse-

quentes das regras de R_d decide qual classe possui maior contagem e a retorna como predição do registro d_l (Linhas 11-19). No caso de predição por regras de associação, os modelos de aprendizagem são atualizados com o registro d_l (Linha 16), como descrito na seção 3.5.

Em caso de empate no esquema de votação para todos os k-itemsets, ou em casos que não são encontradas nenhuma regra de associação em R cujo antecedente case com os k-itemsets de d_l, a predição do registro é passada para os classificadores auxiliares do *ensemble*. A função *GetClass&Sim* na Linha 21 obtém a predição de cada classificador em $\mathscr{C}$ para o registro d_l, assim como o grau de confiança ou similaridade em relação à classificação. Para cada classificador em $\mathscr{C}$, um limite inferior t_{inf} e um limite superior t_{sup} de similaridades também são fornecidos como parâmetros. A função retorna, então, a lista de classes C_d predita pelos classificadores, o número de predições *countUp* com similaridade superior a t_{sup} e o número de predições *countDown* cujas similaridades são inferiores a t_{inf}. Se todos os classificadores predisseram a classe de d_l com similaridades superiores a t_{sup} (Linha 22), então a predição de cada classificador é verificada (Linhas 23-26). Se todas as predições são para a mesma classe, essa classe é retornada como a predição de d_l e o modelo de aprendizagem é atualizado (Linhas 27-30). Se os classificadores não concordam quanto à classe de d_l, esse registro é adicionado à fila Q para ser predita na segunda rodada da fase de teste (Linha 31-34).

Por outro lado, se todos os classificadores predisseram a classe de d_l com similaridades inferiores a t_{inf} (Linha 35), então d_l é predito como pertencente a uma nova classe. Um novo rótulo é criado para a nova classe e o modelo de aprendizagem é atualizado (Linhas 36-38). Caso contrário, o registro é adicionado à fila Q, e vai ser predito na segunda rodada do teste (Linhas 39-42).

Nos dois classificadores auxiliares utilizados nos experimentos deste trabalho, ambos baseados em similaridade, um registro de teste é comparado com os registros de treinamento, e a classe correspondente à *string* com maior similaridade é escolhida como a classe do registro testado.

Em suma, se o registro pode ser classificado por regras de associação ou se todos os classificadores auxiliares concordam quanto à classe do registro e com confiança superior a um limite estabelecido, então a classe do registro é predita e o mesmo é usado na atualização do modelo de aprendizagem. Se todas as predições têm baixa confiança, o registro é predito como pertencente a uma nova classe. Nos demais casos, a predição do registro é adiada para a segunda rodada do teste.

Algoritmo 2: Fase de Teste - Primeira Rodada

Require: Test record d_l
Require: Set of rules R
Require: List of classifiers $\mathscr{C}$
Ensure: The predicted class of d_l or d_l added to queue Q

$S_0 \leftarrow$ Tokenize(d_l)
$m \leftarrow$ Length(S_0)
for $(k \leftarrow 1; k \leq m; k++)$ **do**
  $R_d \leftarrow \emptyset$
  $S_k \leftarrow$ GenItemSets(k, S_{k-1})
  for each k-itemset $it \subseteq S_k$ **do**
    for each $X \rightarrow c \in R$ such that $it = X$ **do**
      $R_d \leftarrow R_d \cup X \rightarrow c$
    end for
  end for
  for each $r \in R_d$ in the form $X \rightarrow c$ **do**
    $c.count++$
  end for
  $pc \leftarrow c_i$ such that $c_i.count > c_j.count \; \forall j \neq i$
  if $pc \neq \emptyset$ **then**
    UpdateModel(R, $\mathscr{C}$, d_l, pc)
    return pc // The predict class of d_l
  end if
end for
// Tie in voting or no rule
$C_d, countUp, countDown \leftarrow$ GetClass&Sim($\mathscr{C}$, d_l)
if $countUp ==$ Length($\mathscr{C}$) **then**
  for each class $c \in C_d$ **do**
    $c.count++$
  end for
  $pc \leftarrow c_i$ such that $c_i.count > c_j.count \; \forall j \neq i$
  if $pc.count ==$ Length(L) **then**
    // Unanimous vote for a class
    UpdateModel(R, $\mathscr{C}$, d_l, pc)
    return pc // The predict class of d_l
  else
    // Record classification postponed
    InsertFIFO(Q, d_l)
  end if
else if $countDown ==$ Length($\mathscr{C}$) **then**
  // Novel class detected
  UpdateModel(R, $\mathscr{C}$, d_l, $newLabel$)
  return $newLabel$ // The predicted class of d_l
else
  // Record classification postponed
  InsertFIFO(Q, d_l)
end if

Exemplo 2. Considere os registros de teste da Figura 3.2. Após a execução do Algoritmo 2 para o registro d_5, sua predição será a classe c_2, predita pela regra de associação r_4. Para o registro d_6, não existem regras no modelo associativo, e a similaridade com qualquer registro de treinamento é baixa. Então esse registro é considerado pertencente a uma nova classe, o rótulo c_4 é criado e retornado como predição de d_6. Esse registro ainda é usado para atualização do modelo de aprendizagem, e adiciona as regras r_5 e r_6 ao classificador associativo. O registro d_7 será predito como pertencente à nova classe criada c_4, através das regras r_5 e r_6, adicionadas ao modelo na fase de teste. Com a adição desse registro ao modelo de aprendizagem, a regra r_7 ainda será criada no classificador associativo. Finalmente, o registro d_8 é predito pelos classificadores auxiliares como pertencente à classe c_3, já que é similar ao registro d_4 dessa classe.

3.4.2 Fase de teste - Segunda rodada

Todas os registros que tiveram suas predições adiadas na primeira rodada da fase de teste são classificados na segunda rodada. Os passos dessa rodada são mostrados no Algoritmo 3. Primeiramente, é verificado se o registro d_l pode ser classificado pelo classificador associativo (Linhas 1-18). Esses passos são os mesmos do Algoritmo 2, com exceção de que o registro não é inserido no modelo de aprendizagem. Note que um registro que não pode ser classificado na primeira rodada do teste por regras de associação agora talvez o seja, já que a atualização do modelo de aprendizagem com outros registros de teste podem ter gerado regras que casem com algum k-item obtido do registro que teve sua predição adiada. Se não é possível realizar a predição com regra de associação, por falta de regras ou empate na votação, a decisão é tomada pelos classificadores auxiliares. A classe predita será aquela indicada pela maioria dos classificadores auxiliares, e em caso de empate, a predição final de d_l será dada por um classificador preferencial. O classificador preferencial é definido pela ordem em que os classificadores aparecem na lista $\mathscr{C}$.

3.5 Atualização do Modelo de Aprendizagem

Cada classificador no *ensemble* pode ter seu modelo atualizado na primeira rodada da fase de teste, com a chamada da função *UpdateModel* do Algoritmo 2 descrito na Seção 3.4.1. A atualização depende da implementação de cada classificador. Abaixo é descrita a atualização do classificador associativo.

Algoritmo 3: Fase de Teste - Segunda Rodada

Require: Test record $d_l \in Q$
Require: Set of rules R
Require: List of classifiers $\mathscr{C}$
Ensure: The predict class of d_l

1: $S_0 \leftarrow$ Tokenize(d_l)
2: $m \leftarrow$ Length(S_0)
3: **for** $(k \leftarrow 1; k \leq m; k++)$ **do**
4: $R_d \leftarrow \emptyset$
5: $S_k \leftarrow$ GenItemSets(k, S_{k-1})
6: **for each** k-itemset $it \subseteq S_k$ **do**
7: **for each** $X \rightarrow c \in R$ such that $it = X$ **do**
8: $R_d \leftarrow R_d \cup X \rightarrow c$
9: **end for**
10: **end for**
11: **for each** $r \in R_d$ in the form $X \rightarrow c$ **do**
12: $c.count++$
13: **end for**
14: $pc \leftarrow c_i$ such that $c_i.count > c_j.count \; \forall j \neq i$
15: **if** $pc \neq \emptyset$ **then**
16: return pc // The predicted class of d_l
17: **end if**
18: **end for**
19: // Tie in voting or no rule
20: $C_d \leftarrow \emptyset$
21: **for each** classifier $\mathscr{C}a \in \mathscr{C}$ **do**
22: $class \leftarrow$ Classify($\mathscr{C}a$, d_l)
23: Insert($class$, C_d)
24: **end for**
25: **for each** class $c \in C_d$ **do**
26: $c.count++$
27: **end for**
28: $pc \leftarrow c_i$ such that $c_i.count > c_j.count \; \forall j \neq i$
29: **if** $pc \neq \emptyset$ **then**
30: return pc // The predict class of d_l
31: **end if**
32: // Class predicted by the preferred classifier among those which tied for the most
33: // voted classes
34: return ClassOfPreferredClassifier()

Sempre que novos registros de dados são usados na atualização do modelo de aprendizagem, novas regras de associação podem ser geradas e outras podem deixar de existir. Novas regras de associação serão geradas se o registro apresentar k-itens que apresentem tanto 100% de confiança quanto atendam ao suporte mínimo. Por outro lado, regras de associação já existentes podem ser eliminadas do modelo, caso deixem de apresentar 100% de confiança ou deixem de

Algoritmo 4: Atualização do Classificador Associativo.

Require: Set of rules R
Require: Training record d_j
Require: Class label c_i

Ensure: Set of rules R and inverted index updated

1: $S_0 \leftarrow$ Tokenize(d_j)
2: **for each** token $t_k \in S_0$ **do**
3: InsertInvertedIndex(t_k, j, c_i)
4: **end for**
5: // The same minimum support *ms* used in the training phase
6: RemoveRulesWithoutSupport(c_i, *ms*)
7: $m \leftarrow$ Length(S_0)
8: **for** $(k \leftarrow 1; k \leq m; k++)$ **do**
9: $S_k \leftarrow$ GenItemSets(k, S_{k-1})
10: **for each** k-itemset $it \subseteq S_k$ **do**
11: **if** SizeOccurrenceList(it) $= 1$ **then**
12: // 100% confidence rule
13: **if** SupportRule($it \rightarrow c_i$) $\geq ms$ **then**
14: **if** $it \rightarrow c_i \notin R$ **then**
15: InsertRule($it \rightarrow c_i$, R)
16: **end if**
17: **end if**
18: RemoveItemSet(it, S_k)
19: **else if** $X \rightarrow c_l \in R$ such that $it = X$ **then**
20: // The rule exists, but it is no longer 100% confidence
21: RemoveRule($it \rightarrow c_l$, R)
22: **end if**
23: **end for**
24: **end for**

atender ao suporte mínimo. Para a implementação da atualização do classificador associativo, é necessário manter a estrutura de dados do índice invertido criado na fase de treinamento, assim como o conjunto de regras inicial.

O Algoritmo 4 descreve os passos da atualização. Primeiro, os *tokens* obtidos da descrição do registro d_j da classe c_i são inseridos no índice invertido (Linhas 1-4). Nessa inserção, se o *token* já existe no índice invertido, o registro de dados é inserido na lista de ocorrências do mesmo. Se o *token* ainda não foi indexado, uma nova entrada é adicionada ao vocabulário, e a lista de ocorrências do *token* é inicializada com o registro de dados. Dessa forma, o modelo de classificação associativa é capaz de incorporar novas *features*, tratando o fenômeno denominado evolução dos atributos.

Então a função *RemoveRulesWithoutSupport* (Linha 7) checa e remove as regras da classe c_i que não mais atendem ao suporte mínimo *ms*. À medida que novos registros são inseridos, o mínimo de registros que um itemset necessita ocorrer aumenta. Isso é eficientemente implementado, já que requer somente a verificação da lista de ocorrências dos *k*-itemsets das regras da classe c_i no índice invertido.

Após isso, os *k*-itemsets do registro são gerados pela função *GenItemSets* e é verificado quando eles formam regras de associação com 100% de confiança para a classe (Linhas 8-26), similarmente ao Algoritmo 1 da fase de treinamento. Se um *k*-itemset forma uma regra de associação com 100% de confiança e possui suporte superior ao mínimo, uma nova regra é adicionada ao modelo, se ainda não existe uma regra com esse *k*-itemset (Linhas 12-18). Adicionalmente, uma regra que já possui 100% de confiança pode perder esse valor com a inserção de novos registros, se esse possui um *k*-itemset que forma uma regra para a classe c_l, diferente da classe c_i. Nesse caso, a regra é removida do classificador associativo (Linha 24).

Note que tanto a inserção quanto a remoção de regras no classifiador associativo eventualmente leva a mudanças nas definições das classes. Conforme já discutido, essas mudanças caracterizam um fenômeno denominado mudança do conceito, e a criação/remoção de regras no classificador associativo trata naturalmente algumas formas de ocorrência desse fenômeno. Ressalta-se que em nenhum momento foi objetivo deste trabalho tratar tal fenômeno ou mesmo identificar a sua ocorrência.

3.6 *Merge* de novas entidades

Em busca de melhorias nos resultados, foi implementada e experimentada uma fase de *merge* para o método, que ocorre após a segunda rodada de teste. Seja C_{new} o conjunto de classes que foram criadas automaticamente em tempo de teste pelo método e $\mathscr{T}_{new}$ o conjunto de registros do incremento classificados como pertencentes a essas novas classes, $\mathscr{T}_{new} \subseteq \Delta\mathscr{T}$. A fase de *merge* do método consiste em tentar agrupar novas classes criadas automaticamente pelo método AssocIER, a partir das descrições dos registros de teste classificados como pertencentes a cada uma delas. O objetivo é tentar agrupar registros de uma mesma nova classe que foram classificadas em duas ou mais novas classes distintas.

Para isso, as descrições dos registros classificados em uma mesma nova classe c_i, $c_i \in C_{new}$ são concatenadas em uma única *string*, de forma que cada nova classe possua apenas um registro de dados. Após a concatenação, uma função de similaridade é usada para calcular

Figura 3.3 – Exemplo de execução da fase adicional para o *merge* de novas classes.

- - - - - - *Conjunto de teste* - - - - - - - - - - - -	*Predição* - - - - - -
d_5: S7 Edge	c_2
d_6: Moto G4	c_4 (nova classe 1)
d_7: Smartphone Moto G4	c_4
d_8: Samsung Galaxy S7	c_3
d_9: Celular Motorola G 4ª Geração	c_5 (nova classe 2)

Fonte: Do autor (2017).

a semelhança entre cada par de registros $d_i \in c_i$ e $d_j \in c_j$, $i \neq j$, e caso a similaridade seja superior a um limiar t_{merge} que precisa ser ajustado, as classes c_i e c_j são agrupadas em uma única nova classe. Assim, todos os registros classificados inicialmente pelo método AssocIER como pertencentes às classes c_i ou c_j passam a ter a mesma predição.

Exemplo 3: Considere os registros de teste da Figura 3.3 e suas respectivas predições após a execução do método AssocIER. Os registros d_5 a d_8 são os mesmos registros de teste da Figura 3.2 e como mostrado pelo Exemplo 2, após a execução da fase de teste a predição dos registros d_5, d_6, d_7 e d_8 são, respectivamente, as classes c_2, c_4, c_4 e c_3, sendo a classe c_4 uma nova classe criada em tempo de predição. O registro d_9, por sua vez, foi identificado como pertencente a segunda nova classe, cujo rótulo criado é c_5. Os registros d_6, d_7 e d_9, classificados como pertencentes a novas classes, são usados na fase adicional do método proposto para avaliar a possibilidade de agrupamento das duas novas classes em uma única nova classe. Para isso, os registros d_6 e d_7, classificados como pertencentes a nova classe c_4, são concatenados e geram o registro "Moto G4 Moto G4 Smartphone". Essa nova descrição é comparada através de uma função de similaridade com os registros da outra nova classe c_5, igualmente concatenados, mas que neste caso possui apenas o registro "Celular Motorola G 4ª Geração". Se a similaridade entre tais registros for superior ao limiar $t_{merge} = 0,8$, por exemplo, os registros das classes c_4 e c_5 passarão a ter a mesma predição e essas classes passarão a ser uma única nova classe.

3.7 Fases da resolução de entidades

Como descrito na Seção 2.1.2, métodos para a resolução de entidades geralmente compreendem as seguintes fases (TALBURT, 2010): preparação dos dados, *blocking*, representação do problema, computação das similaridades, agrupamento dos registros, e *merging*. A seguir é descrita a utilização ou não de cada fase no método proposto:

a) **preparação dos dados:** registros de dados são comumente descritos por um conjunto de vários atributos (*features*) específicos do domínio. Neste trabalho, focou-se em registros descritos por um único atributo textual curto. Registros com mais de um atributo tipicamente textual também podem ser utilizados se forem concatenados em uma única *string*. O pré-processamento nos dados consistiu na remoção de *stopwords* (preposições, conjunções, artigos...), sinais de pontuação e outros;

b) ***blocking:*** o objetivo dessa fase é aumentar a eficiência do método, já que comparar todos os pares de registros tem complexidade quadrática. Com a utilização de classificação multiclasses não são necessárias todas essas comparações. Essa fase não foi usada no método proposto. Ressalta-se com a utilização de algum método de *blocking* a eficiência do *baseline* melhoria, mas para que a comparação ficar justa, *blocking* também deveria ser aplicado ao método proposto. Todavia, como pode ser observado nos resultados obtidos no Capítulo 5, o método proposto já é muito eficiente, mesmo sem a utilização de um algoritmo de *blocking*. Além disso, uma divisão mal feita dos blocos pode levar a perda de eficácia;

c) **representação do problema:** para a utilização de algoritmos de aprendizagem supervisionada, cada registro de dados é representado na forma d_{j,c_i}, em que j é uma *string* e c_i é o rótulo da classe (entidade) ao qual o registro faz referência;

d) **computação das similaridades e agrupamento dos registros:** em abordagens tradicionais para a resolução de entidades, primeiro é calculada a semelhança entre os pares de registros da base de dados e em seguida é utilizado um algoritmo de *clustering* para agrupar registros que fazem referência à mesma entidade. No método proposto, essas fases são compreendidas pelo treinamento dos algoritmos de aprendizagem supervisionada utilizando os registros do conjunto de treinamento, e pela predição à qual entidade os registros do conjunto de teste ou incremento correspondem na fase de teste. Note que após a fase de teste, os conjuntos de registros que fazem referência à mesma entidade já são conhecidos, logo não é necessária a utilização de um algoritmo de *clustering*;

e) ***merging:*** no método proposto, esta fase é opcional, e consiste em tentar agrupar somente novas entidades, isto é, somente entidades que não ocorreram em tempo de treinamento. Essa fase é descrita na Seção 3.6.

3.8 Complexidade computacional

A complexidade computacional do método proposto depende da complexidade dos classificadores que compõem o *ensemble*. O classificador associativo é o mais importante e o único fixo no método, então somente sua complexidade é analisada.

A complexidade de tempo do classificador associativo na fase de treinamento é dominado pelo número de registros a serem utilizados na fase e pelo número de *tokens* em suas descrições. No pior caso, todos os *tokens* em cada descrição precisam ser combinados para gerar os *itemsets*. Seja n o número de registros e m a média de *tokens* nesses registros. Então a complexidade de tempo do classificador associativo é $O(n * 2^m)$. No entanto, o número de *tokens* por *string* não é tão alto em bases de dados reais e não depende do número de registros. Então, o número de *tokens* m pode ser considerado uma constante, e a complexidade de tempo do classificador associativo se torna linear, $O(n)$.

A mesma análise de complexidade pode ser feita para a fase de teste. É importante observar que a busca por uma regra de associação no conjunto R pode ser realizada em $O(1)$ se for utilizada uma tabela *hash* para armazenar as regras.

Com o objetivo de reduzir o tempo de execução, foram implementadas algumas estratégias de poda. A primeira delas consiste em remover do conjunto de *items* aqueles k-itemsets que já geraram alguma regra de associação, evitando combiná-los na próxima iteração do algoritmo (Linhas 10 dos Algoritmos 1 e 4). A segunda estratégia é a função *GenItemSets*, que evita combinar itemsets que não atingem o suporte mínimo. A terceira estratégia consiste em limitar a no máximo 3 o número de itemsets a serem gerados.

4 CONFIGURAÇÃO DOS EXPERIMENTOS

Neste capítulo, são apresentados os software e hardware, as bases de dados e as métricas utilizadas na avaliação do método proposto.

4.1 Software e hardware

Os experimentos deste trabalho foram executados em um servidor HP ProLiant DL 120 G7 com processador Intel Quad-Core Xeon E3-1270 3,4 GHz, 8 MB de cache e 8 GB de memória RAM, com Ubuntu Server 14.04.2 LTS.

Os algoritmos foram implementados em linguagem de programação Java 1.9. Adicionalmente, a biblioteca CoreNLP (MANNING et al., 2014) foi utilizada para o *lemmatization* (BALAKRISHNAN; LLOYD-YEMOH, 2014) dos *tokens* durante o pré-processamento dos registros de dados. Para a implementação do *baseline* descrito na Seção 4.3, foram utilizados algoritmos de aprendizagem de máquina disponíveis no Weka - Waikato Environment for Knowledge Analysis (HALL et al., 2009), versão 3.7.9.

4.2 Bases de dados

Nos experimentos, foram utilizadas bases de dados compostas por referências a entidades reais descritas por dados textuais curtos. As seguintes bases de dados foram utilizadas:

a) **Cora Ref**[1] ***dataset*:** esta base de dados é composta por referências bibliográficas para alguns trabalhos científicos, faz parte do projeto Cora (MCCALLUM et al., 2000; MCCALLUM; NIGAM; UNGAR, 2000);

b) **Printer *dataset*:** base de dados cujos registros são descrições de impressoras recuperados no Google Shopping[2], criada por Pereira et al. (2011);

c) **Computer Science Publication Venue - (PVAF) *dataset*:** base de dados gerada por Pereira, Silva e Esmin (2014), possui dados referentes a títulos de veículos de publicações coletados em repositórios digitais e *sites* de instituições que divulgam trabalhos da área

[1] http://www.cs.umass.edu/ mccallum/data/ cora-refs.tar.gz
[2] http://www.google.com/shopping

de Ciência da Computação, tais como o ACM Digital Library[3], a DBLP[4] e o Qualis Capes[5];

d) **UOL *dataset*:** coletados no site UOL Shopping[6] por Oliveira (2014) e Oliveira e Pereira (2017). Essa base foi dividida em três coleções distintas. Na base de dados denominada *UOL-electronics* as ofertas de produtos se referem a produtos eletrônicos, a saber, produtos da categoria celular, telefone, eletrônicos, informática e eletrodomésticos. A base *UOL-non-electronics* é composta por ofertas de produtos das categorias perfumaria e cosméticos, acessórios de moda e joias, brinquedos e jogos, esporte, *fitness*, bebês e crianças. Na base de dados denominada *UOL-books*, não avaliada neste projeto, todos os registros são de ofertas de livros;

e) **Abt-Buy *datasets*:** bases de dados compostas por descrições de produtos obtidas dos sites Abt[7] e Buy[8], foram utilizadas por Köpcke, Thor e Rahm (2010);

f) **Amazon-Google *datasets*:** bases de dados compostas por descrições de produtos coletadas dos sites Amazon[9] e Google Shopping, também foram utilizadas por Köpcke, Thor e Rahm (2010) na avaliação de abordagens para resolução de entidades;

g) **DBLP-ACM *datasets*:** base de dados utilizada por Köpcke, Thor e Rahm (2010), é composta por informações bibliográficas de publicações científicas em periódicos e conferências relacionadas à Ciência da Computação. Tais publicações estão disponíveis nas bibliotecas digitais ACM[10] e DBLP[11];

h) **DBLP-Scholar *datasets*:** também utilizada por Köpcke, Thor e Rahm (2010), nesta base de dados os registros de entidades são referências bibliográficas de publicações relacionadas à Ciência da Computação. Parte dos registros foi obtida a partir da DBLP e o restante coletado automaticamente pelos autores a partir da máquina de busca do Google Scholar[12];

[3] http://dl.acm.org/
[4] http://www.informatik.uni-trier.de/ ley/db/
[5] http://qualis.capes.gov.br/webqualis/principal.seam
[6] http://www.shopping.uol.com.br/
[7] http://www.abt.com/
[8] http://www.buy.com/
[9] http://www.amazon.com/
[10] http://dl.acm.org/
[11] http://dblp.uni-trier.de/
[12] https://scholar.google.com

i) **Name Disambiguation *dataset*:** essa base de dados contém registros de nomes de autores extraídos da biblioteca digital DBLP, e contém vários grupos de autores ambíguos. Sua versão original foi criada por (HAN et al., 2004) e foi usada por (FERREIRA et al., 2010), (FERREIRA et al., 2014) e (SANTANA et al., 2015) com algumas alterações.

Algumas informações sobre as bases de dados utilizadas nos experimentos estão sumarizadas na Tabela 4.1. Para a base de dados *Name Disambiguation* as estatísticas são apresentadas por grupos ambíguos na Tabela 4.2. Como pode ser observado nessas tabelas, a maioria das bases de dados possui um número de entidades (classes) distintas grande e com poucos registros, o que deixa a tarefa de resolução de entidades mais complexa. A coluna *Tokens* (variação) apresenta o número mínimo e máximo de *tokens* presentes nas descrições dos registros. Na coluna *Tokens* (média), é apresentada a média de *tokens* por registro.

4.3 *Baselines*

O método proposto foi comparado com uma abordagem tradicional não incremental baseada em aprendizagem de máquina para resolução de entidades discutidas por Köpcke, Thor e Rahm (2010). Nessa abordagem, a resolução de entidades acontece em duas fases. Para a primeira delas, é necessário um conjunto de pares de registros devidamente rotulados como pertencentes à mesma entidade (*matching*) ou não (*non-matching*). Esses pares são utilizados para treinar um algoritmo de aprendizagem com o objetivo de derivar um modelo de decisão. Na segunda fase, o modelo gerado na primeira fase é utilizado para determinar se pares de registros, não rotulados, são da mesma entidade ou não.

Nesse método, foi utilizado o SVM (LibSVM..., 2009; CORTES; VAPNIK, 1995) como algoritmo de aprendizagem, que apresentou os melhores resultados na avaliação reportada por Köpcke, Thor e Rahm (2010), e as seguintes funções de similaridade: Cosseno (SALTON; MCGILL, 1983), Jaccard (JACCARD, 1901), Distância de Edição (LEVENSHTEIN, 1966), Sorensen Dice (SØRENSEN, 1948) e Trigrams (KONDRAK, 2005), individualmente ou combinadas como em (BILENKO; MOONEY, 2003). Nesse caso, cada componente no vetor de atributos é representado pela similaridade entre dois registros, calculada usando uma ou mais funções de similaridade.

Tabela 4.1 – Estatísticas das bases de dados.

Base de Dados	Registros	Entidades	Registro por entidade (variação)	Registros por entidade (média)	*Tokens* distintos	*Tokens* (variação)	*Tokens* (média)
Cora-Ref	1880	191	1-236	9.84	1935	6-68	18.12
Printer	2168	157	2-29	13.81	1898	1-18	7.86
UOL-electronics	10229	2341	2-53	4.37	11556	1-40	14.12
UOL-non-electronics	26640	5299	2-91	5.03	12441	1-32	8.01
PVAF	2252	663	3-13	3.40	2231	2-24	7.36
Abt-Buy	2177	1081	1-3	2.01	3988	2-30	8.41
Amazon-Google	2663	1363	1-6	1.95	3136	2-37	7.12
DBLP-ACM	4448	2223	2-4	2.0	11662	5-56	19.10
DBLP-Scholar	7559	2350	1-37	3.22	13938	4-57	17.54

Fonte: Do autor (2017).

Tabela 4.2 – Estatística da base de dados *Name Disambiguation*.

Grupo Ambíguo	Registros	Entidades (Autores)	Registros por entidade (variação)	Registros por entidade (média)	*Tokens* distintos	*Tokens* (variação)	*Tokens* (média)
agupta	576	26	2-108	22.15	2103	8-35	16.88
akumar	243	14	1-95	17.36	994	4-42	16.2
cchen	798	60	2-108	13.3	2531	5-48	17.09
djohnson	368	15	2-181	24.53	1258	5-34	15.3
jmartin	112	16	2-21	7.0	668	7-28	16.38
jrobinson	171	12	2-44	14.25	815	8-32	16.05
jsmith	904	29	1-175	31.17	3044	6-50	17.21
ktanaka	280	10	2-108	28	1159	9-44	17.83
mbrow	153	13	3-41	11.77	751	8-35	16.16
mjones	260	13	2-59	20.0	1227	2-46	18.57
mmiller	412	12	2-193	34.33	1914	8-48	19.4

Fonte: Do autor (2017).

A efetividade de uma abordagem que usa aprendizagem de máquina depende de um conjunto de dados para treinamento expressivo e preferencialmente pequeno (KÖPCKE; THOR; RAHM, 2010). A similaridade entre um par de registros deve exceder um limiar t, para garantir que o conjunto de treinamento não seja dominado por pares de registros *non-matching* triviais. Em Köpcke, Thor e Rahm (2010), os autores recomendam uma porcentagem mínima de pares *mathcing* ou *non-matching*. Após extensiva experimentação, os autores asseguram que $t = 0,4$ e 40% de pares *matching* ou *non-matching* é uma configuração padrão confiável. Esta configuração foi usada na experimentação deste trabalho, com a seleção aleatória de pares para o treinamento. O conjunto de treinamento foi composto de 20 a 10000 pares, dependendo do tamanho da base de dados.

Os resultados reportados neste trabalho para cada base de dados são os melhores alcançados, após a busca dos melhores parâmetros para o SVM, avaliação de diversos tamanhos de conjunto de treinamento e diferentes combinações das funções de similaridade. Os resultados são a média de 10 execuções usando validação cruzada nos mesmos conjuntos de treinamento e teste em que o método AssocIER foi executado.

Para a base *Name Disambiguation*, os resultados obtidos pelo método AssocIER foram comparados com os apresentados em Santana et al. (2015). Esses algoritmos não foram executados pela falta de acesso aos algoritmos e indisponibilidade de tempo para a implementação dos mesmos.

4.4 Métricas de avaliação

A avaliação é uma importante etapa no desenvolvimento de um novo método, pois permite determinar a qualidade dos resultados obtidos e comparar esses resultados com outras abordagens. A seguir são descritas as métricas utilizadas na avaliação do método proposto neste trabalho.

Uma das métricas utilizadas na avaliação da efetividade do método proposto foi a métrica K (LAPIDOT, 2002), utilizada na avaliação de outras soluções para a resolução de entidades. Seja N o número de registros no conjunto de teste, R o número de classes de referência manualmente criadas, q o número de classes criadas automaticamente pelo método, $n_{i,j}$ o número de elementos da classe de referência i que pertencem a classe automaticamente criada j e n_i o número de elementos da classe i. A métrica K combina a Pureza Média da Classe (PMC) e a Fragmentação Média da Classe (FMC). PMC é definida pela Equação 4.1, e avalia a pureza

das classes, isto é, avalia se cada classe contém somente registros da mesma classe, comparando as classes criadas automaticamente pelo algoritmo e as criadas manualmente. Por outro lado, FMC avalia se as classes manualmente criadas foram fragmentadas entre as classes automaticamente indicadas pelo método e é definida pela Equação 4.2. Assim, quanto menos classes automáticas contêm registros da mesma classe manualmente criada, menos fragmentada é a classe de referência. A métrica K, por sua vez, é uma média geométrica que combina PMC e FMC, e é dada pela equação 4.3.

$$PMC = \frac{1}{N} \sum_{i=1}^{q} \sum_{j=1}^{R} \frac{n_{i,j}^2}{n_i} \tag{4.1}$$

$$FMC = \frac{1}{N} \sum_{j=1}^{R} \sum_{i=1}^{q} \frac{n_{i,j}^2}{n_j} \tag{4.2}$$

$$K = \sqrt{PMC \times FMC} \tag{4.3}$$

Também foram utilizadas métricas calculadas a partir do número de pares de entidades corretamente identificados pelo classificador. A métrica pF_1 (F_1 par a par), tradicionalmente utilizada na avaliação de métodos para resolução de entidades, é descrita pela Equação 4.4 e é uma média harmônica que combina a precisão par a par (pp - Equação 4.5) e a revocação par a par (pr - Equação 4.6) (KÖPCKE; RAHM, 2008). Para o cálculo de pp e de pr são considerados os pares de registros que o método indicou como referências à mesma entidade (M_p) e os pares de registros que correspondem à mesma entidade de acordo com o conjunto de referência (M_a).

$$pF_1 = \frac{2\ pp \times pr}{pp + pr} \tag{4.4}$$

$$pp = \frac{|M_a \cap M_p|}{|M_p|} \tag{4.5}$$

$$pr = \frac{|M_a \cap M_p|}{|M_a|} \tag{4.6}$$

Adicionalmente, a métrica F_1 descrita por Albertini e Mello (2007) e o erro mensurado como proposto por Masud et al. (2013) foram utilizados para avaliar o método apresentado

quanto à qualidade da detecção de novas entidades na fase de teste. Essas métricas são descritas a seguir.

A métrica F_1 (Equação 4.9) (ALBERTINI; MELLO, 2007) é uma variação da métrica de avaliação para problemas de classificação F_1 tradicional. Para o seu cálculo, são consideradas a precisão e revocação, conforme as Equações 4.7 e 4.8, respectivamente. Nessas equações, T_{nov} é o número de registros de dados detectados como pertencentes a novas entidades que realmente são novidades, D_{nov} é o número de registros detectados como novas entidades pelo classificador e N_c é o número total de registros pertencentes a novas entidades.

$$P = \frac{T_{nov}}{D_{nov}} \tag{4.7}$$

$$R = \frac{T_{nov}}{N_c} \tag{4.8}$$

$$F_1 = \frac{2PR}{P+R} \tag{4.9}$$

O erro do método foi mensurado de acordo com Masud et al. (2013), cujas métricas foram propostas para a avaliação da classificação de *data streams*. Seja $Enovexist$ o total de registros de novas classes que foi predito pertencer a uma classe existente, $Eexistnov$ o total de registros de classes existentes que foi predito pertencer a uma nova classe e $Eother$ o total de registros classificados incorretamente entre as classes já existentes. Seja ainda N o total de registros e V o total de registros de novas classes. Então a porcentagem de registros de novas classes classificados incorretamente como pertencentes a classes já existentes é dada pela Equação 4.10. A porcentagem de registros de classes já existentes classificada incorretamente como pertencente a novas classes é dada pela Equação 4.11. Já o erro global é dado pela Equação 4.12. Entretanto, essas medidas consideram a tarefa de classificação como binária, isto é, todas as novas classes observadas são consideradas uma única classe.

$$M_{new} = \frac{100 \times Enovxist}{V} \tag{4.10}$$

$$F_{new} = \frac{100 \times Eexistnov}{N-V} \tag{4.11}$$

$$Error = \frac{100 \times (Enovexist + Eexistnov + Eother)}{N} \tag{4.12}$$

4.5 Configuração do método

Nos experimentos com o método AssocIER, foi usado um *ensemble* de classificadores composto pelo classificador associativo e dois classificadores baseados em similaridade: um usando o coeficiente de similaridade de Jaccard e outro usando a similaridade do Cosseno. Esse último implementado usando a função de *ranking* proposta por Persin (1994).

Os limiares usados no módulo para detecção de novas classes para ambos os classificadores baseados em similaridade foram $tInf = 0.6$ e $tSup = 0.8$. Eles foram escolhidos após experimentação com a técnica *grid-search*, usando validação cruzada sobre os dados de treinamento. A experimentação mostrou que esses valores são uma configuração padrão efetiva em todas as bases de dados testadas.

O suporte mínimo para o classificador associativo varia de 0,0 a 0,3, exceto para a base *Cora-Ref*, cujo suporte mínimo utilizado foi 0,8. Os valores usados para cada base são apresentados na Tabela 4.3, que também apresenta os limiares utilizados no classificador baseado em Cosseno. As características dos dados dão algumas dicas quanto ao suporte mínimo a ser utilizado. Para bases de dados como a *Cora-Ref*, com *strings* mais longas, mais registros por classe e *strings* muito similares nas classes, um suporte mínimo alto contribui para a eliminação de *tokens* ruidosos, que provavelmente gerariam regras de associação ruins. Por outro lado, para a base de dados *Name Disambiguation*, em que os registros em cada classe não representam réplicas, mas são citações para trabalhos do mesmo autor, o suporte mínimo igual a zero leva a melhores resultados, já que evita a eliminação de *tokens* que ocorrem em apenas alguns registros.

Os parâmetros *cINS* e *cADD* do classificador baseado no Cosseno são utilizados pelo algoritmo de Persin na filtragem de registros. Em vez de calcular a similaridade de todos os *tokens* como em uma função de similaridade tradicional, o algoritmo de Persin calcula tal similaridade um *token* por vez. Assim, durante o processamento de um *token* tk, uma similaridade parcial $sim_{q,d,tk}$ do registro de teste d e o registro de treinamento q. Se a similaridade parcial do registro de teste d e um registro de treinamento q é superior a *cINS*, então esse registro de treinamento é suficiente importante para ser um registro candidato à determinação da classe de teste. Se o registro q já está presente no conjunto de registros de treinamento relevante, a

similaridade parcial $sim_{q,d,tk}$ é adicionada ao acumulador desse registro. Em caso contrário, o registro q é adicionado ao conjunto de registros relevantes. Por outro lado, se $sim_{q,d,tk}$ é menor que *cINS* e maior que *cADD*, o registro de treinamento q é considerado não importante o bastante, mas pode influenciar a ordem final do ranqueamento. Nesse caso, se o registro q já faz parte do conjunto de registros relevantes, a similaridade $sim_{q,d,tk}$ é adicionada ao acumulador de q. Se $sim_{q,d,tk}$ é menor que *cADD*, a informação do *token* tk é considerada não importante, e o próximo registro de treinamento e processado. Os parâmetros *cINS* e *cADD*, $cINS \geq cADD$, são recalculados sempre que um novo *token* é avaliado. O parâmetro *cINS* é um parâmetro de inserção, e permite selecionar como candidatos somente aqueles registros de treinamento relevantes. Já o parâmetro *cADD* permite ignorar o cálculo de similaridades parciais. Ambos contribuem para a redução do tempo de execução do método.

Neste trabalho, o ajuste desses parâmetros também foi feito com a técnica *grid-search* e validação cruzada sobre os dados de treinamento. Primeiramente foi ajustado o parâmetro *cINS*, com o objetivo de reproduzir os resultados de uma função de ranqueamento que usa o Cosseno como similaridade base mas que compara um registro de teste com todos os registros de treinamento. Em seguida, foi buscado o valor de *cADD* que minimizasse o tempo de execução.

Tabela 4.3 – Parâmetros utilizados na execução dos experimentos.

Base de dados	**Classificador Associativo**		**Classificador baseado no Cosseno**	
	Suporte Mínimo	Tamanho máximo do *itemset* (passo)	cINS	cADD
Cora Ref	0.8	3	1.0	0.0
Printers	0.3	3	1.0	0.0
PVAF	0.2	3	1.7	0.0
UOL-eletronic	0.3	3	1.0	0.0
UOL-non-eletronic	0.3	3	1.0	0.0
UOL-book	0.3	3	1.0	0.0
Abt-Buy	0.5	3	0.9	0.2
Amazon-Google	0.5	3	1.9	0.0
DBLP-ACM	0.3	3	1.0	0.1
DBLP-Scholar	0.4	3	1.0	0.1
Name Disambiguation	0.0	3	0.7	0.0

Fonte: Do autor (2017).

Nos experimentos foram utilizados 50% dos dados como conjunto de treinamento e 50% como incremento, exceto quando explicitado de outra forma. Cada base de dados foi aleatoriamente dividida em 10 *folds*, e cada experimento foi executado 10 vezes, cada um deles

usando 5 *folds* como conjunto de treinamento e os 5 *folds* restantes como incremento. Os resultados apresentados neste trabalho são a média de 10 execuções. A mesma estratégia foi aplicada no método tradicional não incremental usado como *baseline*, de modo que em cada execução foram utilizadas as mesmas amostras usadas no treinamento e teste do AssocIER. As comparações são estatisticamente significantes com intervalos de confiança de 95% usando o teste t.

4.6 Outros classificadores no *ensemble*

Além dos classificadores baseados na similaridade do Cosseno e na similaridade do Jaccard, outros classificadores também foram testados na composição do *ensemble* utilizado no método proposto. Dentre eles, vale mencionar um classificador semelhante aos que utilizam a similaridade do Cosseno e do Jaccard, em que a similaridade entre registros utiliza conceitos da lógica *fuzzy* (ZADEH, 1965), e os algoritmos de classificação SVM e Naive Bayes. O classificador com lógica *fuzzy* não foi utilizado porque os resultados obtidos foram bem inferiores aos obtidos pelos classificadores baseados no Cosseno e Jaccard. Já o SVM e o Naive Bayes foram desconsiderados porque manter esses classificadores atualizados se mostrou extremamente ineficiente. Em todos esses algoritmos, os registros de dados precisam ser transformados em vetores de *features*, em que para cada *feature* é atribuído um peso, por exemplo o *TF-IDF*, e o custo benefício da atualização desses pesos não se mostrou interessante. Vale lembrar que no classificador baseado no Cosseno os registros de dados são mapeamentos nesses vetores, mas durante a experimentação deste trabalho decidiu-se não atualizar os pesos das *features*. Tal decisão foi tomada porque esse processamento é custoso e também não levou a melhorias significativas nos resultados.

4.7 Ferramenta de suporte à análise de resultados

Com o objetivo de facilitar a análise dos resultados, foi desenvolvida uma ferramenta para visualização e análise dos mesmos. Nesta seção, é apresentada a ferramenta e as tecnologias utilizadas em seu desenvolvimento.

4.7.1 A ferramenta

A ferramenta está disponível em https://joaoasilva.shinyapps.io/resultsApp/ e consiste essencialmente na plotagem de um conjunto de gráficos a partir dos arquivos de saída gerados

pelo método AssocIER e pelo *baseline*, arquivos com as métricas de aviação e medidas de tempo. Adicionalmente, para cada base de dados são apresentadas informações como o seu número de registros, o número de entidades distintas, média de *tokens* nos registros e outras. A Figura 4.1 apresenta uma das páginas da ferramenta. Pela barra de navegação superior é possível selecionar as análises já implementadas, cujos gráficos foram plotados da seguinte forma:

Figura 4.1 – Uma das páginas da ferramenta de suporte à análise de dados.

Fonte: Do autor (2017).

a) comparação da abordagem para resolução de entidades tradicional usada como *baseline* e o método AssocIER (seção *Incremental ER X Traditional ER*). Para esta análise foram plotadas a métrica pF_1 obtida por ambos os métodos e os seus respectivos tempos de treinamento;

b) avaliação do método AssocIER (seção *Incremental ER*): nessa página foi plotado um conjunto de métricas que avalia diferentes aspectos do método. A página foi dividida em 4 abas distintas, para análise dos resultados apresentados no Capítulo 5. Na aba *Experiment 1* são apresentados os resultados obtidos pelo método AssocIER em sua configuração padrão, assim como os resultados obtidos nas seguintes configurações alternativas: atualização do modelo de aprendizagem somente com registros de novas classes, AssocIER com passo adicional para agrupamento de novas classes e AssocIER

com pré-processamento adicional dos registros de dados. Na aba *Experiment 2* são apresentados os resultados obtidos pelo método AssocIER quando o conjunto de teste é formado apenas por registros de novas classes. Na aba *Experiment 3* são apresentados os resultados obtidos quando o conjunto de teste não possui nenhum registro pertencente a novas classes e na aba *Experiment 4* é possível analisar a influência do tamanho do conjunto de treinamento nos resultados;

c) na seção *Name Disambiguation* são apresentados os resultados obtidos pelo método AssocIER no problema de desambiguação de nomes de autores;

d) análise dos resultados obtidos pelo *baseline seção Traditional ER*: nessa seção foram plotados os resultados obtidos pelo método tradicional para resolução de entidades usado como *baseline* usando diferentes algoritmos de classificação e algumas funções de similaridade, assim como combinações dessas;

e) na seção *Data Analysis* são apresentadas estatísticas sobre as bases de dados, as mesmas apresentadas nas Tabelas 4.1 e 4.2. Além disso, é plotado um histograma com a distribuição dos registros entre as classes.

Em todas as análises do método AssocIER, os valores usados para a geração dos gráficos também são apresentados em forma de tabela.

4.7.2 Tecnologias utilizadas

A ferramenta foi desenvolvida utilizando **Shiny** (CHANG et al., 2015). Shiny é um pacote para o ambiente de desenvolvimento RStudio[13] que permite desenvolver aplicações Web iterativas utilizando a linguagem R (R Core Team, 2015). Esse pacote consiste em um conjunto de funções em sintaxe da linguagem R que substituem o HTML, CSS e JavaScript. Algumas das características do pacote Shiny são sumarizadas a seguir:

a) desenvolvimento da aplicação em poucas linhas de código, totalmente iterativa sem necessidade de JavaScript;

b) *front-end* da aplicação pode ser desenvolvida integralmente em linguagem R, mas também é possível utilizar HTML, CSS e JavaScript;

c) tema padrão baseado em Bootstrap[14] (Framework JavaScript-CSS);

[13] https://www.rstudio.com/
[14] http://getbootstrap.com/

d) livre, sob a licença GPLv3 (General Public License versão 3).

Uma aplicação desenvolvida utilizando Shiny pode ser executada localmente com o RStudio ou em um navegador, e pode ser executada *online* através de sua implantação em um servidor Shiny. No caso deste projeto, a aplicação foi implantada em um servidor disponibilizado pela comunidade responsável pelo RStudio (shinyapps.io[15]). A plataforma possui alguns planos pagos, mas a ferramenta desenvolvida neste projeto foi disponibilizada no plano grátis, cujos recursos foram suficientes para o propósito da ferramenta, mesmo que tenha uma série de limitações.

[15] https://www.shinyapps.io/

5 RESULTADOS E DISCUSSÃO

Neste capítulo é apresentado e discutido um conjunto de experimentos executados para avaliar diferentes aspectos da abordagem proposta.

5.1 Comparação com o *baseline*

Os resultados obtidos pelo AssocIER e pelo *baseline* (TER) são apresentados na Tabela 5.1. Para o método tradicional de resolução de entidades, é apresentada somente a métrica pF_1, uma vez que o método apenas encontra os pares de registros de uma mesma entidade. Um algoritmo adicional seria necessário para agrupar todos os registros da mesma entidade. Adicionalmente, é apresentada a média de novas classes, de acordo com o conjunto de registros utilizado como incremento. Os resultados para as demais bases de dados (Abt-Buy, Amazon-Google, DBLP-ACM, DBLP-Scholar e Name Disambiguation) são apresentados em seções específicas (Seção 5.8 e Seção 5.10).

A comparação da métrica pF_1 obtida por ambos os métodos permite observar que o método AssocIER apresenta resultados melhores que o *baseline* nas três bases de dados cujos dados são descrições de ofertas de produtos (valores em negrito), com ganhos de até 149%. A identificação das entidades nas bases de dados *UOL-electronics* e *UOL-non-electronics* é particularmente difícil, uma vez que possuem um grande número de classes, a maioria delas com poucos registros. A maioria dos registros nas bases de dados *Printer* e *UOL-electronics* possui algum código implícito para o produto, como o código "J3680" no registo "HP Offcejet J3680 printer". A identificação desse tipo de código é uma característica importante do classificador associativo usado no método AssocIER, que gera regras de associação com esses *tokens* na fase

Tabela 5.1 – Métricas K e pF_1 (e seus intervalos de confiança) obtidas pelo método AssocIER e mética pF_1 obtida pelo *baseline* TER. A coluna novas classes apresenta o número de novas classes existentes no incremento, de acordo com o conjunto de referência.

Base de Dados	AssocIER		TER	Novas
	K	pF_1	pF_1	Classes
Printer	0.943 ± 0.005	$\mathbf{0.931 \pm 0.007}$	0.374 ± 0.003	4
UOL-electronics	0.867 ± 0.002	$\mathbf{0.804 \pm 0.004}$	0.390 ± 0.006	551
UOL-non-electronics	0.751 ± 0.001	$\mathbf{0.563 \pm 0.010}$	0.246 ± 0.004	668
Cora-Ref	0.905 ± 0.009	0.899 ± 0.012	0.900 ± 0.008	63
PVAF	0.768 ± 0.010	0.591 ± 0.023	0.605 ± 0.010	253

Fonte: Do autor (2017).

de treinamento/atualização do modelo e posteriormente as usa na classificação de outros registros. Ressalta-se que esse tipo de código também é utilizado no *baseline*. Por outro lado, a base *UOL-non-electronics* não apresenta nenhum tipo de código implícito. Essa característica torna a resolução dessa base de dados mais difícil, mas ainda assim AssocIER apresenta resultados melhores que o *baseline*.

A base de dados *PVAF* possui muitos registros com acrônimos ou siglas dos nomes de veículos de publicações. As classes nessa base de dados também possuem poucas registros, e muitas vezes há uma grande diferença na descrição de registros da mesma classe. Um exemplo são os registros "SC Conference" e "Conference for High Performance Computing, Networking, Storage and Analysis". Bases de dados com informações bibliográficas como a base *Cora-Ref* geralmente são mais fáceis de resolver que bases de dados com informações de produtos, fato também verificado por Köpcke, Thor e Rahm (2010).

A métrica K no método AssocIER é melhor que a métrica pF_1. Isso ocorre porque as classes com apenas um registro no conjunto de teste não formam nenhum par e, portanto, não contribuem para a métrica pF_1, mesmo quando os registros dessas classes são corretamente classificados. A maior diferença entre as métricas ocorre justamente nas bases com poucos registros por classe, isto é, nas bases *UOL-electronics*, *UOL-non-electronics* e *PVAF*.

É importante observar que neste experimento os registros no incremento são tanto de classes que já ocorreram na fase de treinamento quanto de classes que emergem na fase de teste. Assim, bases de dados com poucos registros por classe tendem a ter mais registros de novas classes na fase de teste, o que também pode ser observado na Tabela 5.1. Note que as 253 novas classes que emergem na fase de teste são 38% das classes da base de dados *PVAF*. Na bases *UOL-electronics* e *UOL-non-electronics* essa porcentagem é 23,5% e 12,6% respectivamente. Todos os casos demonstram a habilidade do método proposto em identificar grandes números de classes emergentes, assim como corretamente classificar registros de classes já conhecidas pelo modelo de decisão.

Para avaliar a eficiência do método AssocIER, foi comparado o seu tempo de execução com o tempo de execução do *baseline*. O tempo de execução, em segundos, para as fases de treinamento e teste de ambos os métodos é apresentado na Tabela 5.2 e é a média de 10 execuções. Como pode ser observado, AssocIER leva mais tempo para treinamento que o método tradicional em todas as bases de dados, exceto na base *Uol-non-electronics*. O tempo gasto nessa fase do AssocIER é composto pela geração das regras de associação e criação das estru-

Tabela 5.2 – Tempo de treinamento e teste em segundos (e seus intervalos de confiança) no método AssocIER e no *baseline* TER. Os menores tempos estão destacados em negrito.

Base de Dados	AssocIER		TER	
	Treinamento	Teste	Treinamento	Teste
Printer	0.55 ± 0.09	$\mathbf{1.26 \pm 0.09}$	$\mathbf{0.39 \pm 0.04}$	32.31 ± 0.62
UOL-electronics	32.14 ± 0.25	$\mathbf{99.19 \pm 0.66}$	$\mathbf{2.03 \pm 0.14}$	575.33 ± 10.72
UOL-non-electronics	$\mathbf{21.17 \pm 0.25}$	$\mathbf{274.48 \pm 1.78}$	40.90 ± 4.33	15774.61 ± 159.18
Cora-Ref	1.09 ± 0.15	$\mathbf{4.87 \pm 0.36}$	$\mathbf{0.23 \pm 0.035}$	14.39 ± 0.16
PVAF	$\mathbf{0.28 \pm 0.08}$	$\mathbf{1.46 \pm 0.16}$	0.59 ± 0.07	22.70 ± 0.45

Fonte: Do autor (2017).

turas dos outros dois classificadores baseados em similaridade que constituem o *ensemble*, e pelo cálculo do TF-IDF no classificador baseado no Cosseno. O *baseline* é mais rápido nessa fase porque o número de pares usados no treinamento é pequeno, sendo os melhores resultados obtidos com até 5000 pares.

Por outro lado, o método proposto é muito mais eficiente na fase de teste, mesmo que nela sejam realizadas a detecção de novas classes, atualização do modelo de decisão e alguns registros tenham sua predição adiada. Como pode ser observado na Tabela 5.3, a maioria dos registros do incremento geralmente é classificada com regras de associação, o que leva tempo linear em função do número de registros desse conjunto. Somente na base de dados *UOL-non-electronics* a maioria dos registros é classificada pelos classificadores auxiliares. Como já mencionado, a ausência de um código implícito na descrição dos registros dessa base de dados dificulta a classificação. Em seu trabalho, Oliveira (2014) reporta que apenas 69,22% dos registros dessa base podem ser classificados por regras de associação, com validação cruzada de 10 *folds*, 9 deles usados como treino e o outro como teste. No experimento deste trabalho, a por-

Tabela 5.3 – Média de registros classificados pelo classificador associativo e pelos classificadores auxiliares em cada rodada do teste.

Base de Dados	Primeira Rodada			Segunda Rodada	
	Novas classes	Classificador associativo	Classificadores auxiliares	Classificador associativo	Classificadores auxiliares
Printer	21	955	24	3	81
UOL-electronics	563	3105	400	51	992
UOL-non-electronics	532	3850	2836	40	6060
Cora-Ref	30	358	427	2	122
PVAF	167	513	33	36	377

Fonte: Do autor (2017).

centagem de registros classificados com regras de associação é 29,21%. O método tradicional, por sua vez, necessita avaliar todos os pares formados pelos registros de teste, o que demanda tempo quadrático em função do número de registros do conjunto e faz o *baseline* não escalável para grandes volumes de dados.

A eficiência de métodos para resolução de entidades pode ser melhorada com técnicas que reduzem o número de comparações entre registros, como *blocking* e *canopies* (ELMAGARMID; IPEIROTIS; VERYKIOS, 2007). No entanto, tais técnicas podem não ser efetivas em dados cujas referências são semiestruturadas ou não estruturadas, casos em que qualquer mal alinhamento nos dados levará à perda de qualidade nos resultados.

Para Gruenheid, Dong e Srivastava (2014), são dois os objetivos da resolução incremental de entidades: (i) o método incremental deve obter resultados similares a métodos em *batch* e (ii) a resolução incremental de entidades deve ser muito mais rápida que a execução do método em *batch*, especialmente quando o incremento é pequeno. Pelos resultados obtidos, o método AssocIER provou-se muito mais rápido e com resultados mais efetivos que o TER, que executa em modo *batch*. Portanto, o método proposto atende aos objetivos de um método para resolução incremental de entidades conforme sugerido por Gruenheid, Dong e Srivastava (2014).

A solução apresentada por Gruenheid, Dong e Srivastava (2014) mantém incrementalmente um grafo de similaridades para os registros de dados e realiza *clustering* incremental desse grafo. No entanto, como mostrado pelos resultados obtidos com o *baseline* nos experimentos deste trabalho, uma abordagem que considera similaridade entre pares de registros não apresenta bons resultados em dados textuais como os aqui verificados. Ressalta-se que na experimentação com o *baseline*, foram utilizadas diversas configurações distintas para o método proposto por Köpcke, Thor e Rahm (2010), com vários algoritmos de aprendizagem diferentes e extensiva combinação de funções de similaridade na formação dos registros. Ademais, conforme mostrado por Köpcke, Thor e Rahm (2010), o método tradicional com uso de aprendizagem de máquina produz resultados melhores que outras abordagens tradicionais. Assim, a utilização de um algoritmo de *clustering* usando tais similaridades também não levaria a bons resultados.

5.2 Fase adicional e outros pré-processamentos nos dados

Em busca de melhorias nos resultados, o AssocIER foi experimentado em duas outras configurações. Na primeira, foi executada a fase adicional para agrupamento de novas classes

como descrito na Seção 3.6. Na segunda, além do pré-processamento padrão dos registros (remoção de *stopwords*, sinais de pontuação e transformação de caixa) também foi aplicada a técnica *stemming* e *lemmatization*.

Os resultados obtidos com as configurações alternativas são apresentados na Tabela 5.4, e mostram que os processamentos adicionais não alteram significativamente ambas as métricas K e pF_1. No caso de uso da fase adicional para o *merge* das novas classes criadas automaticamente pelo AssocIER, ainda há um pequeno acréscimo de tempo na fase de teste. Na base *UOL-non-electronics*, esse acréscimo é de aproximadamente 2%. Há ainda a dificuldade de se ajustar o parâmetro adicional t_{merge}. Para os resultados apresentados na Tabela 5.4, a fase adicional para o método AssocIER foi executada utilizando a similaridade do Cosseno e $t_{merge} = 0,8$, isto é, duas ou mais novas classes foram agrupadas em uma única nova classe se as similaridade entre os registros das classes foi superior a 0,8.

Os resultados da coluna AssocIER-3 na Tabela 5.4 foram obtidos com a adição de *stemming* (BALAKRISHNAN; LLOYD-YEMOH, 2014) ao pré-processamento dos registros, e também não alterou os resultados. Isso ocorre porque os *tokens* que são afetados por tais mudanças já são naturalmente eliminados durante a geração de regras de associação através da avaliação do suporte da regra gerada. Adicionalmente, tal técnica depende do conhecimento do vocabulário específico de algum idioma, geralmente o Inglês. As bases *UOL-electronics* e *UOL-non-electronics* foram coletadas em um Web site do Brasil, e a maioria dos *tokens* nos registros são do Português, para a qual não se conhece a existência de um método que use vocabulário do idioma. Então, esperava-se que a técnica não alterasse os resultados nessas bases. O *lemmatization* (BALAKRISHNAN; LLOYD-YEMOH, 2014) é uma técnica similar ao *stemming*, para qual há a possibilidade de se utilizar o vocabulário referente ao Português (Portugal). Ainda assim, os resultados não foram alterados nas bases *UOL-electronics* e *UOL-non-electronics*.

5.3 Avaliação do auto treinamento

Neste experimento, foi avaliada a capacidade de auto treinamento incremental do método proposto, isto é, sua capacidade de agregar novos registros de dados ao modelo de decisão. A habilidade foi comparada com três outras configurações alternativas do método, descritas a seguir:

Tabela 5.4 – Métricas K e pF_1 obtidas em diferentes configurações do método: método AssocIER padrão (coluna AssocIER); método AssocIER com fase adicional para agrupamento de novas classes (coluna AssocIER-2); e método AssocIER com a utilização de *stemming* no pré-processamento dos registros (coluna AssocIER-3).

Bases de Dados	AssocIER		AssocIER-2		AssocIER-3	
	K	pF_1	K	pF_1	K	pF_1
Printer	0.943	0.931	0.942	0.931	0.945	0.935
UOL-electronics	0.867	0.804	0.867	0.800	0.867	0.801
UOL-non-electronics	0.751	0.563	0.758	0.563	0.758	0.563
Cora-Ref	0.905	0.899	0.908	0.900	0.904	0.899
PVAF	0.786	0.591	0.770	0.610	0.758	0.585

Fonte: Do autor (2017).

a) AssocIER-2: método AssocIER com detecção de novas classes, e atualização incremental do modelo de predição somente com registros classificados como pertencentes a novas entidades na primeira rodada do teste;

b) AssocIER-3: método AssocIER sem detecção de novas classes e com atualização incremental do modelo;

c) AssocIER-4: método AssocIER sem detecção de novas classes e sem atualização do modelo.

Os experimentos com as configurações alternativas do método foram executados na mesma configuração do experimento da Seção 5.1, usando 50% de cada base de dados como treino e 50% como teste. Assim como naquele experimento, os resultados apresentados são a média de 10 execuções.

Os resultados deste experimento são apresentados na Tabela 5.5, e demostram que o auto treinamento é um importante recurso do método proposto. Novamente, a maior diferença dos resultados pode ser observada nas bases de dados que possuem poucos registros por classe (*UOL-electronics* e *PVAF*), e evidencia a importância de agregar conhecimentos dos novos registros de dados ao modelo de decisão, sejam esses registros de novas classes ou não.

Ainda é possível observar pelos resultados da Tabela 5.5 que a segunda configuração alternativa (AssocIER-3), que apenas atualiza o modelo de decisão, apresenta resultados inferiores à terceira configuração alternativa (AssocIER-4), na qual não ocorre a atualização do modelo. Ambas as configurações classificam incorretamente todos os registros de novas classes, e no caso da segunda configuração esse erro é propagado com a atualização do modelo. A atualização do classificador associativo com informações incorretas pode causar a geração

Tabela 5.5 – Avaliação da capacidade de auto treinamento e detecção de nova entidades em diferentes configurações do método AssocIER: com auto treinamento e atualização com todas as classes (coluna AssocIER); auto treinamento e atualização somente com novas classes (coluna AssocIER-2); sem detecção de novas classes e auto treinamento (coluna AssocIER-3); e sem detecção de novas classes e nem auto treinamento (coluna AssocIER-4). Os melhores resultados estão destacados em negrito.

Base de Dados	AssocIER		AssocIER-2		AssocIER-3		AssocIER-4	
	K	pF_1	K	pF_1	K	pF_1	K	pF_1
Printer	**0.941**	**0.930**	0.938	0.927	0.928	0.913	0.933	0.921
UOL-electronics	**0.867**	**0.804**	0.827	0.744	0.748	0.642	0.752	0.651
UOL-non-electronics	**0.751**	**0.564**	0.732	0.545	0.706	0.514	0.714	0.539
Cora-Ref	**0.905**	**0.899**	0.875	0.879	0.841	0.856	0.857	0.878
PVAF	**0.765**	**0.590**	0.690	0.422	0.580	0.280	0.586	0.306

Fonte: Do autor (2017).

de regras de associação ruins ou mesmo a eliminação de regras boas, e assim compromete a qualidade das predições feitas pelo método. Adicionalmente, nos classificadores baseados em similaridade, a adição de um registro com rótulo equivocado ao conjunto de treinamento agrava o cenário. Por outro lado, na terceira configuração alternativa o método se comporta como um classificador não incremental. Assim, mesmo que erre a predição de todos os registros de novas entidades esse erro não é propagado com a atualização do modelo.

5.4 Novas classes e classes conhecidas pelo modelo de decisão

O objetivo deste experimento é avaliar o método AssocIER em dois cenários distintos: (i) o conjunto de teste é formado por apenas registros de classes de novas classes, isto é, de classes que não ocorreram no conjunto de treinamento; e (ii) não há nenhum registro de nova classe no teste, isto é, todos os registros do conjunto de teste são de classes que já ocorreram no conjunto de treinamento.

Nos dois casos foi realizada validação cruzada com 10 *folds*. Para isso, a base de dados foi aleatoriamente dividida em 10 partes, cada uma com registro de 10% das classes da base de dados, e 9 delas usadas como conjunto de treinamento e a parte restante como incremento. Assim, para o experimento somente com novas classes, em cada execução da validação cruzada o incremento é composto por 10% de classes da base de dados e nenhuma delas ocorreu no conjunto de treinamento. No experimento sem novas classes, os registros no incremento pertencem a 10% das classes da base de dados, mas todas as classes ocorreram no conjunto de treinamento. Os resultados são a média das 10 execuções da validação cruzada. Em ambos os

casos o método AssocIER foi executado com o mecanismo de detecção de novas classes e auto treinamento.

Os resultados obtidos no cenário em que apenas registros de novas classes são utilizados como conjunto de teste são apresentados na Tabela 5.6. As métricas K e pF_1 evidenciam que o método AssocIER é eficaz mesmo na presença de um número grande de classes. Uma vez que todos os registros no conjunto de teste são de novas classes, a métrica $pF_{1_{new}}$, que considera somente os pares formados por registros de novas classes, deveria ser igual à pF_1 tradicional. Mas isso não ocorre porque alguns registros são classificados incorretamente como pertencentes a classes já conhecidas pelo modelo de decisão ou mesmo incorretamente entre as novas classes detectadas.

A métrica F_1, que avalia somente o número de registros de novas classes corretamente identificadas, também demonstra o bom desempenho do método AssocIER na maioria das bases de dados. O pior resultado ocorreu na base *UOL-non-elctronics*, que como já discutido é particularmente difícil de resolver devido ao grande número de classes com poucos registros. Adicionalmente, os registros dessa base de dados não apresentam um código implícito para o produto, tipo de *token* com alto potencial de geração de boas regras de associação. Mais um agravante é a similaridade entre registros de classes distintas, como por exemplo os registros "Tênis Nike Zoom Structure+ 16" e "Tênis Nike Zoom Structure+ 17". Se uma das classes ocorreu no treino e a outra emergiu na fase de teste, é provável que o método classifique o registro de teste incorretamente, seja porque encontrou regras de associação, mas estas eram ruins, ou porque a similaridade entre os registros é muito alta. As métricas M_{new} e F_{new}, que

Tabela 5.6 – Métricas obtidas pelo método AssocIER com incremento formado apenas por registros de novas classes. Nessa avaliação, a porcentagem de registros de classes já conhecidas incorretamente classificados como novas classes (coluna F_{new}) é zero em todas as bases de dados. Como todos os registros são de novas classes, o erro global (*error*) é a própria porcentagem de registros de novas classes classificados incorretamente como pertencente a classes já conhecidas pelo modelo de aprendizagem (coluna M_{new}). Os menores erros estão destacados em negrito.

Bases de Dados	K	pF_1	$pF_{1_{new}}$	F_1	M_{new}	F_{new}	*Error*
Printer	0.915	0.895	0.861	0.967	0.055	0.0	**0.055**
UOL-electronics	0.878	0.799	0.650	0.830	0.230	0.0	**0.230**
UOL-non-electronics	0.828	0.740	0.470	0.540	0.522	0.0	0.522
Cora-Ref	0.912	0.860	0.380	0.624	0.370	0.0	0.370
PVAF	0.864	0.766	0.565	0.703	0.350	0.0	0.350

Fonte: Do autor (2017).

medem o erro do método, evidenciam esse fato. Como o incremento é formado apenas por registros de novas classes, F_{new} é igual a zero em todas as bases de dados, isto é, nenhum registro de classes já conhecidas pelo modelo foi incorretamente classificado como pertencente a uma nova classe. Por outro lado, a métrica M_{new} mostra a quantidade de registros de novas classes incorretamente classificados como pertencentes a classes já conhecidas pelo modelo, e é a principal fonte de erros. Para o erro global, também são acrescentados aqueles registros de classes já conhecidas pelo modelo de decisão incorretamente classificados, e neste experimento também é igual a zero em todas as bases de dados.

O número de novas classes neste experimento é grande. Nas bases *UOL-electronics* e *UOL-non-electronics* por exemplo, o número de novas classes é em média 234 e 530, respectivamente. Propostas de classificadores com detecção de novas classes encontradas na literatura, como as apresentadas por Faria, Carvalho e Gama (2016) e Masud et al. (2013), são capazes de detectar um número muito pequeno de classes (3 ou 4). Esses métodos ainda foram propostos para dados estruturados e a detecção de novas classes somente acontece se existem muitos registros na mesma região do espaço, características muito distintas das bases de dados utilizadas neste trabalho.

Os resultados usando incremento com todos os registros de classes já conhecidas pelo modelo de decisão são apresentados na Tabela 5.7. As métricas $pF_{1_{new}}$ e F_1 não são apresentadas porque são igual a zero em todas as bases de dados, já que nenhuma nova classe emerge na fase de teste. Pelo mesmo motivo, o erro M_{new} também é igual a zero. Neste cenário, o erro é com-

Tabela 5.7 – Resultados obtidos pelo método AssocIER com conjunto de teste formado apenas por registros de classes conhecidas pelo modelo de decisão. As métricas $pF_{1_{new}}$ e pF_1 não são apresentadas porque são igual a zero em todas as bases de dados, já que não há nenhum registro no incremento que pertence a novas classes. O erro global neste experimento (coluna *Error*) é composto pela porcentagem de registros de classes já conhecidas pelo modelo de decisão classificados incorretamente como pertencentes a novas classes (coluna F_{new}). Ao erro global, ainda é adicionada a porcentagem de registros de classes já conhecida pelo modelo mas que foi incorretamente classificada entre tais classes. Os menores erros estão destacados em negrito.

Bases de Dados	K	pF_1	M_{new}	F_{new}	*Error*
Printer	0.974	0.959	0.0	0.015	**0.040**
UOL-electronics	0.969	0.962	0.0	0.080	0.171
UOL-non-electronics	0.932	0.889	0.0	0.042	0.343
Cora-Ref	0.852	0.727	0.0	0.0	**0.035**
PVAF	0.904	0.716	0.0	0.101	0.277

Fonte: Do autor (2017).

posto pela porcentagem de registros incorretamente classificados como pertencentes a novas classes e pela porcentagem de registros classificados incorretamente entre as classes já conhecidas pelo modelo de decisão. Mesmo que todas as classes já tenham ocorrido no conjunto de treinamento, o módulo de detecção de novas classes eventualmente identifica equivocadamente uma nova classe. O método proposto erra menos nas bases de dados *Printer* e *Cora-Ref*, que são bases com menos classes e cujos registros são relativamente mais fáceis de classificar. O maior erro ocorre na base de dados *Uol-non-electronics*, devido ao grande número de classes presente nessa base de dados, muitas delas com poucos registros, mas também devido à ausência de um *token* que identifica unicamente cada entidade distinta.

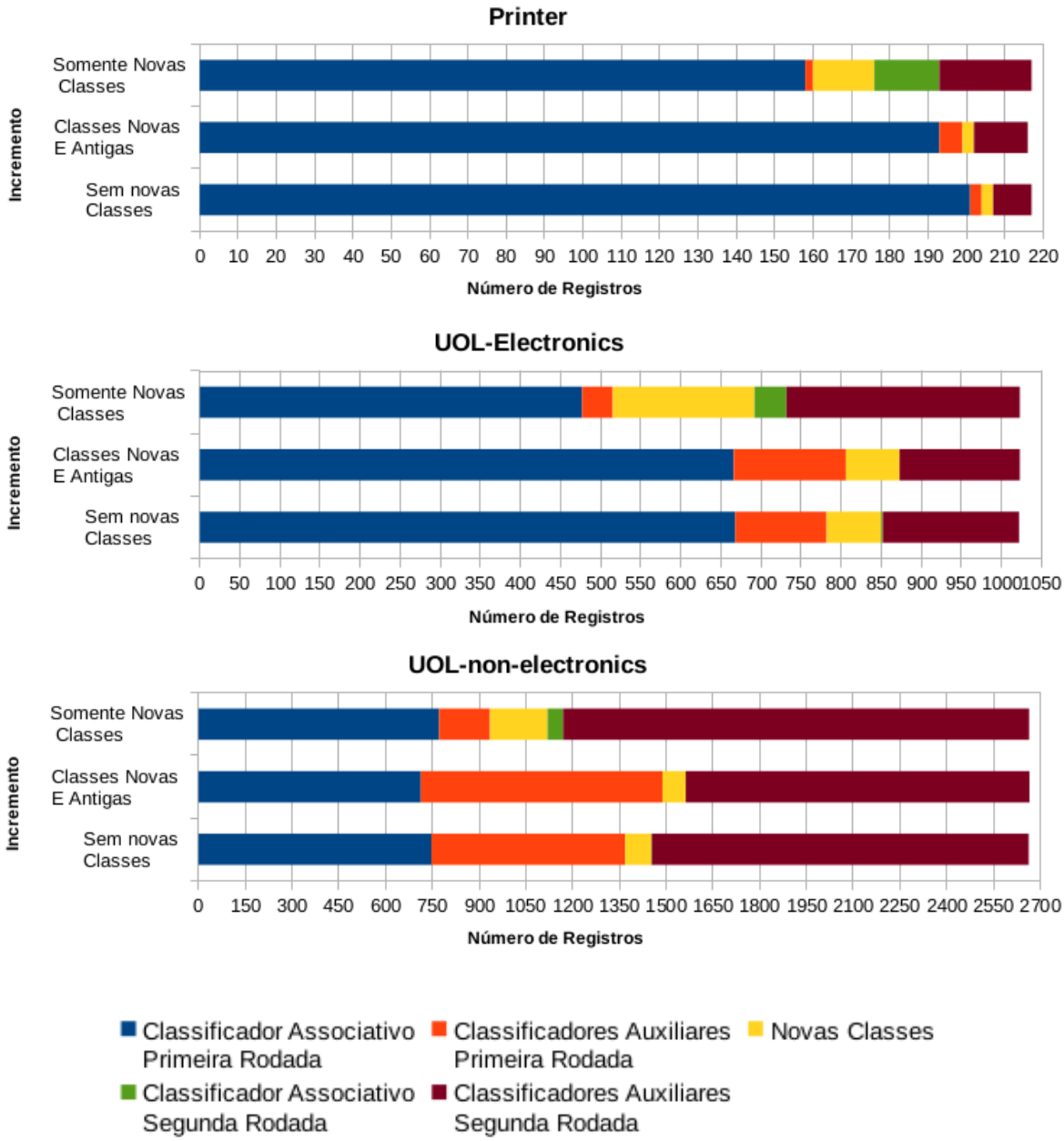
Na Tabela 5.8 é apresentada uma comparação dos resultados obtidos pelo método AssocIER em três cenários: quando o incremento não possui registros de novas classes, quando o incremento possui tanto registros de novas classes quanto de classes já conhecidas pelo modelo de decisão e quando o incremento possui apenas registros de novas classes. O caso em que o incremento possui tanto registros de novas classes quanto de classes já conhecidas pelo modelo de decisão é um caso de dificuldade média, e as métricas obtidas deveriam ficar entre o caso mais fácil (quando o incremento não possui registros de novas classes) e o caso mais difícil (quando o incremento possui apenas registros de novas classes). Mas isso ocorre apenas nas bases de dados *Printer* e *UOL-electronics*, bases de dados em que a maioria dos registros é classificada por regras de associação do classificador associativo, como pode ser observado na Figura 5.1. Nessas bases de dados, a presença de códigos implícitos nas descrições dos registros favorece a geração de regras de associação boas.

Tabela 5.8 – Resultados obtidos pelo método AssocIER quando o incremento não possui registros de novas classes, quando o incremento possui tanto registros de novas classes quanto de classes já conhecidas pelo modelo de decisão e quando o incremento possui apenas registros de novas classes.

Base de Dados	Sem novas classes		Classes novas e antigas		Somente novas classes	
	K	pF_1	K	pF_1	K	pF_1
Printer	0.974	0.959	0.971	0.928	0.915	0.895
UOL-electronics	0.969	0.962	0.976	0.797	0.878	0.799
UOL-non-electronics	0.932	0.889	0.887	0.561	0.828	0.740
Cora-Ref	0.852	0.727	0.946	0.912	0.912	0.860
PVAF	0.904	0.716	0.927	0.756	0.864	0.766

Fonte: Do autor (2017).

Figura 5.1 – Média de registros classificados pelo classificador associativo e pelos classificadores auxiliares em cada rodada do teste nas bases de dados *Printer*, *Uol-electonics*, e *UOL-non-electronics*.



Fonte: Do autor (2017).

Já na base de dados *Uol-non-electronics* o melhor valor da métrica pF_1 ocorre quando não há nenhum registro de novas classes no incremento e seu pior valor ocorre no caso médio. Como pode ser observado na Figura 5.1, nessa base de dados o número de registros classificados por regras de associação é baixo em todos os casos, e é ainda pior quando o incremento possui tanto registros de classes novas quanto de classes já conhecidas pelo modelo de deci-

são. Isso ocorre porque nesse caso a geração do classificador associativo na fase de treinamento acaba eliminando regras de associação que potencialmente classificariam alguns registros do incremento corretamente. Na ausência dessas regras de associação, a maioria dos registros é classificada pelos classificadores auxiliares, que também erram bastante devido à alta semelhança de registros de classes distintas. Por outro lado, quando o incremento só possui registros de novas classes, a detecção dessas e o auto treinamento contribuem para que mais registros sejam classificados com regras de associação. Mais uma observação importante que pode ser notada na Figura 5.1 é o fato de que mesmo quando não há registros de novas classes no incremento, o método AssocIER se equivoca e detecta novas classes. Ainda assim todos os registros que tiveram sua classificação adiada para a segunda rodada são classificados pelos classificadores auxiliares, ou seja, a atualização do modelo de decisão não é influenciada pelo erro na detecção de novas classes.

Na base de dados *Cora-Ref*, o melhor valor da métrica pF_1 ocorre no caso médio, isto é, quando o incremento é formado por registros de classes novas e também por registros de classes já conhecidas pelo modelo de decisão. Como pode ser observado na Figura 5.2, é justamente nesse caso que o método AssocIER classifica o mínimo de registros com regras de associação, mas como demonstrado por Köpcke, Thor e Rahm (2010), em bases de dados com informações bibliográficas abordagens baseadas em similaridades funcionam bem. Assim, os classificadores auxiliares melhoram consideravelmente os resultados nessas bases. O pior resultado nessa base ocorre quando o incremento é formado apenas por registros de classes já conhecidas pelo modelo de decisão, caso em que os classificadores auxiliares mais erram nas predições.

Na base de dados *PVAF*, os valores da métrica pF_1 quando o incremento é formado por registros de novas classes e de classes já conhecidas pelo modelo de decisão e quando todos os registros são de novas classes são iguais, e são melhores que quando os registros do incremento são todos de classes já conhecidas. Na Figura 5.3 é possível notar que mesmo quando não há registros de novas classes no incremento o mecanismo de detecção de novas classes se equivoca, e a detecção de novas classes erroneamente junto a atualização do modelo de aprendizagem faz com que alguns registros sejam classificados incorretamente por regras de associação na segunda rodada do teste. Considerando que os registros identificados como pertencentes a novas classes e os classificados pelos classificadores auxiliares são a maioria, percebe-se que a maior fonte de erro nessa base de dados é decorrente da identificação equivocada de novas classes e

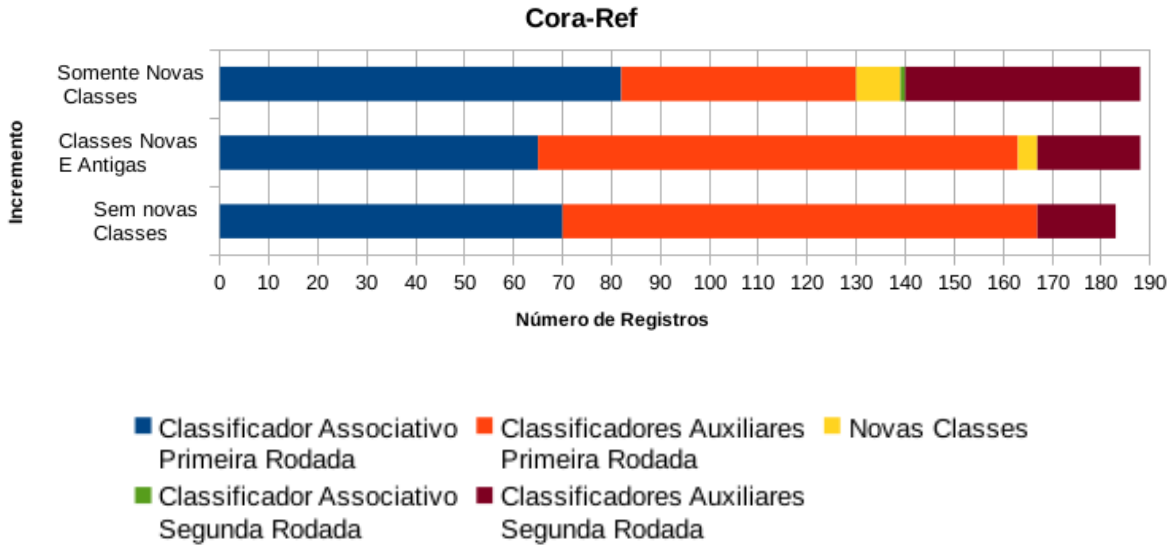
pela classificação dada pelos classificadores auxiliares. Em outras palavras, quando é possível a classificação de um registro usando regras de associação, quase sempre a predição é correta. Mais uma vez, é importante ressaltar que a maioria das classes nessa base de dados possui apenas dois ou três registros, e muitas vezes a descrição de registros da mesma classe è muito diferente.

5.5 Influência do tamanho do conjunto de treinamento

Neste experimento, foi avaliada a efetividade do método AssocIER em cenários em que os dados de treinamento são escassos. Os registros do incremento são os mesmos usados no experimento da Seção 5.1 e o conjunto de treinamento foi aleatoriamente dividido em 10 partes, de forma que cada parte contém 10%, 20%,...,100% dos registros desse conjunto. Assim, quando são utilizados 100% dos dados de treinamento, o experimento é o mesmo da Seção 5.1. Ademais, quanto menor o tamanho do conjunto de treinamento, maior o número de classes que emergem na fase de teste. Este experimento também foi executado sem nenhum dado de treinamento. O método foi executado 10 vezes com cada fração do conjunto de treinamento e os resultados apresentados são a média das execuções.

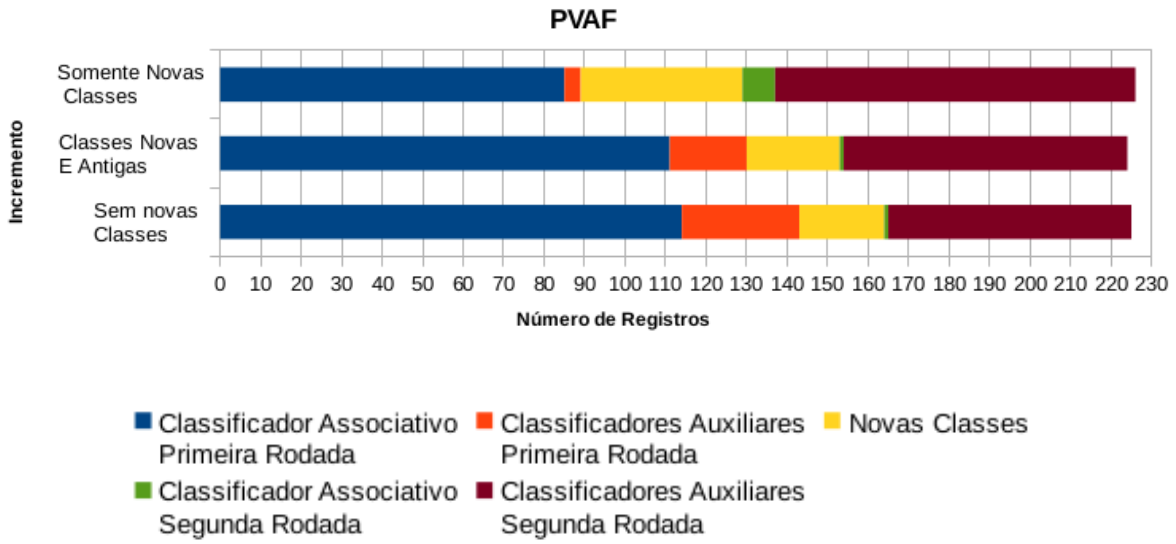
Os gráficos da Figura 5.4 apresentam as métricas K e pF_1 em cada base de dados. O método AssocIER é capaz de resolver o conjunto de teste mesmo quando nenhum dado está

Figura 5.2 – Média de registros classificados pelo classificador associativo e pelos classificadores auxiliares em cada rodada do teste: base de dados *Cora-Ref*.



Fonte: Do autor (2017).

Figura 5.3 – Média de registros classificados pelo classificador associativo e pelos classificadores auxiliares em cada rodada do teste: base de dados *PVAF*.



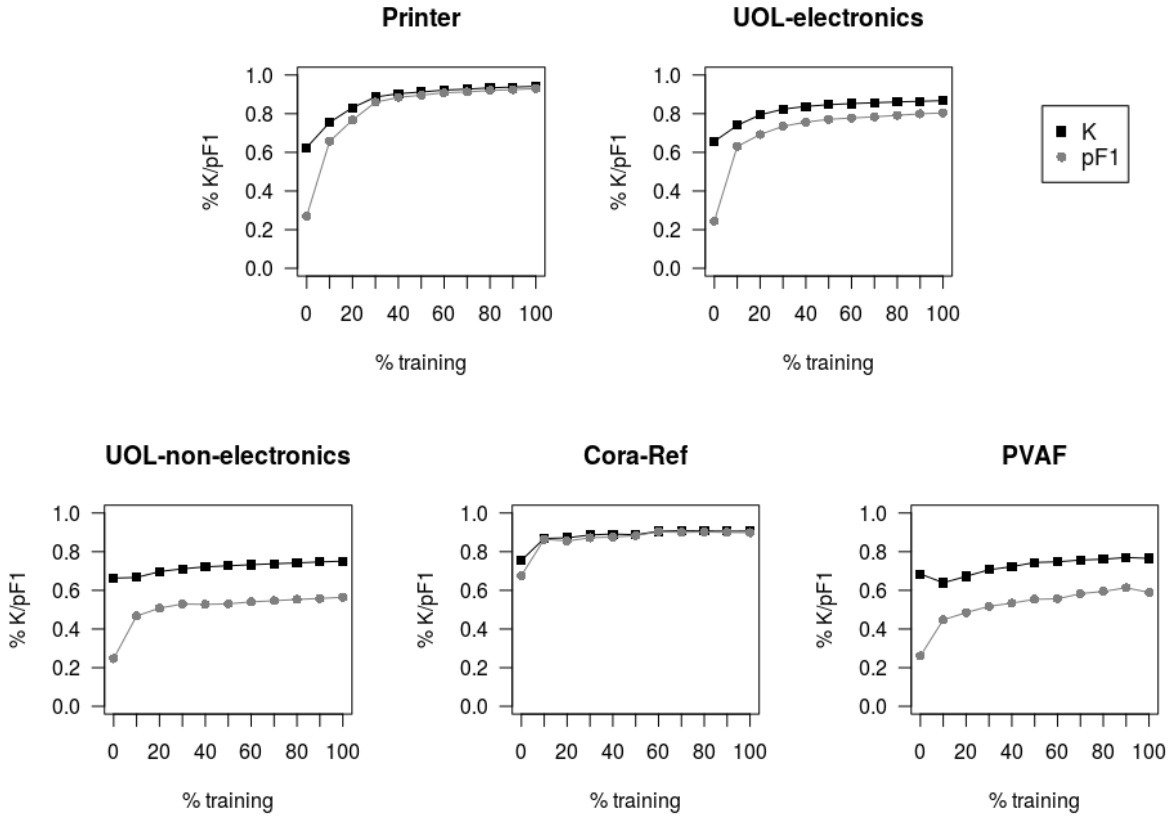
Fonte: Do autor (2017).

disponível para a geração de um modelo de decisão inicial. Nesse caso todas as classes do conjunto de teste são novas classes e o modelo de decisão é totalmente gerado incrementalmente na fase de teste. De fato, mesmo que apenas 10% dos dados de treinamento sejam utilizados, AssocIER apresenta resultados superiores ao *baseline* nas bases de produtos de lojas *online*. Os gráficos da Figura 5.4 ainda permitem observar que a utilização de frações com mais de 40% dos dados de treinamento apresentam resultados muito semelhantes, e ressalta que o método proposto também apresenta bons resultados em cenários que poucos registros estão disponíveis para treinamento.

5.6 Comparação com o algoritmo não incremental proposto por Oliveira (2014) e Oliveira e Pereira (2017)

Para permitir a comparação do método proposto com a versão não incremental apresentada por Oliveira (2014) e Oliveira e Pereira (2017), utilizou-se validação cruzada estratificada, isto é, cada base de dados foi dividida em 10 partes, mas com a garantia de que cada classe fosse proporcionalmente representada em cada parte. Essa divisão faz com que pelo menos um registro de cada classe ocorra no conjunto de treinamento e pelo menos um registro de cada classe ocorra no conjunto de teste, e é necessário porque para o algoritmo não incremental sem detecção de novas classes, se um registro de alguma classe ocorrer no conjunto de teste, então

Figura 5.4 – Métricas K e pF_1 obtidas pelo método AssocIER em cenários com dados para treinamento escassos e mesmo sem nenhum dado para treinamento.



Fonte: Do autor (2017).

pelo menos um registro da mesma classe deve ter ocorrido no conjunto de treinamento. Os experimentos foram executados 10 vezes, usando 9 partes da base de dados como conjunto de treinamento e uma como conjunto de teste.

Na Tabela 5.9 são apresentados os resultados do método AssocIER com auto treinamento mas sem detecção de novas classes e do método base proposto por Oliveira (2014) e

Tabela 5.9 – Resultados obtidos pelo método AssocIER e pelo método proposto por Oliveira (2014) e Oliveira e Pereira (2017) (coluna Método base) .

Base de Dados	AssocIER		Método base	
	K	pF_1	K	pF_1
Printer	0.968	0.900	0.968	0.902
UOL-electronics	0.956	0.669	0.956	0.673
UOL-non-electronics	0.886	0.465	0.887	0.472
Cora-Ref	0.942	0.920	0.943	0.923
PVAF	0.963	0.055	0.962	0.053

Fonte: Do autor (2017).

Oliveira e Pereira (2017) (coluna Método base). Ambos os métodos apresentam resultados semelhantes. Note que nesse cenário em que todas as classes são conhecidas em tempo de treinamento, as regras de associação nos classificadores associativos de ambos os modelos gerados são as mesmas, e a criação/remoção de regras com o auto treinamento com registros do conjunto de teste no classificador incremental não chega a mudar de fato as definições das classes. Todavia, nesse cenário, o método base é mais interessante, já que a fase de teste é mais eficiente por dois motivos: primeiro, a fase de teste do método base utiliza apenas um classificador auxiliar, enquanto o método AssocIER utiliza dois classificadores; e segundo, no método base não ocorre a atualização do modelo. Mais uma vez é importante pontuar a diferença entre as métricas K e pF_1, diferença decorrente do baixo número de registros em algumas classes. De fato, a maior diferença é observada nas bases *UOL-non-electonics* e *PVAF*, e neste experimento, a maioria das classes no conjunto de teste possui apenas um registro, o que leva a baixos valores da métrica pF_1.

5.7 Experimento com a base de dados *Name Disambiguation*

O problema de desambiguação de nomes de autores é um caso específico de resolução de entidades, em que as entidades são autores distintos em um conjunto de referências bibliográficas. Cada referência é composta pelo menos por uma lista com o nome dos autores, o título da publicação e o título do veículo de publicação. Assim, os objetivos são (i) encontrar o conjunto de autores (pessoas) distintos e (ii) atribuir cada referência ao autor correspondente (correto). Existem diversas propostas para a resolução desse problema e um bom estudo sobre pode ser verificado em Ferreira, Gonçalves e Laender (2012). Neste experimento, foi avaliada a efetividade do método AssocIER na solução da desambiguação de nomes de autores. Nesse caso, cada grupo de referências do mesmo autor corresponde a uma classe.

Um dos melhores resultados encontrados para o problema na literatura é apresentado por Santana et al. (2015). Neste experimento foi utilizada uma das bases de dados usada pelos autores (*Name Disambiguation*). Foram utilizadas as mesmas métricas de avaliação e a mesma proporção de dados para treinamento e teste - 50% para treinamento e 50% de teste, em 10 execuções. Todavia, o método NC proposto pelos autores assim como os *baselines* utilizados por eles não foram executados nas mesmas partições de dados que o método AssocIER. Assim, não é possível estabelecer uma comparação direta e estatisticamente confiável com os resultados

apresentados naquele trabalho. Então apenas é apresentada uma noção do resultado obtido pelo método AssocIER.

A Tabela 5.10 apresenta as métricas K e pF_1 obtidas pelo método AssocIER, pelo método proposto por Santana et al. (2015) (coluna NC) e três *baselines* usados por esses autores (SLAND, SVM e NB). O método proposto por Santana et al. (2015) apresenta os melhores resultados (em negrito) e o método AssocIER apresenta resultados superiores ou iguais aos três *baselines* para a métrica K.

Tabela 5.10 – Métricas K e pF_1 obtidas pelo método AssocIER e pelos *baselines* na base de dados *Name Disambiguation*. O método NC foi proposto por Santana et al. (2015) e os métodos SLAND, SVM e NB foram usados como *baselines* por esses autores.

Método	K	pF_1
AssocIER	0.879	0.762
NC	**0.919**	**0.919**
SLAND	0.878	0.869
SVM	0.777	0.702
NB	0.711	0.616

Fonte: Do autor (2017).

Mesmo que o método AssocIER não explore características do problema de desambiguação de nomes, ele obteve resultados promissores na solução do problema. O método apresentado por Santana et al. (2015) usa heurísticas que exploram características específicas do domínio do problema, enquantoAssocIER é um método genérico. Esses resultados motivam a avaliação futura de classificadores específicos na composição do *ensemble*, junto ao classificador associativo, em busca de uma solução específica para tal problema.

O algoritmo SLAND é um método supervisionado com auto treinamento (VELOSO et al., 2012) que utiliza um classificador associativo *lazy*. Nesse algoritmo, dado um registro d, o autor mais provável para tal registro é inferido através da confiança da regra de associação $X \rightarrow a_i$, em que o antecedente da regra possui somente atributos de r_i. SLAND gera as regras de associação no momento em que a citação é apresentada ao método de desambiguação. Adicionalmente, o método é capaz de agregar novos exemplos ao conjunto de treinamento durante o processo de desambiguação, usando predições confiáveis e detecção de autores que previamente não ocorreram no conjunto de treinamento.

Para o uso do SVM, Santana et al. (2015) consideram cada autor distinto como uma classe e realiza o treinamento do classificador para essa classe. Para isso, cada citação é representada por um vetor de *features*, em que cada atributo é descrito usando o esquema de pesos

TF-IDF. Os autores usam a abordagem uma classe X demais classes para realizar a classificação multiclasses com o SVM. No método NB é utilizado um modelo naive Bayes e alguns registros da base de dados são usados como exemplos de treinamento a fim de se estimar os parâmetros em tal modelo. Então, a probabilidade de cada autor a_i gerar uma citação d é calculada usando a regra naive Bayes e a citação d é atribuída ao autor que possui a maior probabilidade de produzir a citação.

5.8 Resultados em outras bases de dados

Nesta seção, são apresentados os resultados obtidos pelo método AssocIER nas bases de dados *Abt-Buy*, *Amazon-Google*, *DBLP-ACM*, e *DBLP-Scholar*. As bases de dados foram divididas em 10 partes, e em cada execução da validação cruzada foram usadas partes como conjunto de treinamento e a parte restante como conjunto de teste. No experimento com essas bases de dados usou-se a proporção 90%-10% na validação cruzada com o objetivo de verificar se o método tradicional não incremental descrito por Köpcke, Thor e Rahm (2010) e implementado para ser usado como *baseline* reproduz os resultados reportados por esses autores. Os resultados obtidos pelo método AssocIER e pelo *baseline* são apresentados na Tabela 5.11, e são a média de 10 execuções. Adicionalmente, são apresentados os resultados reportados em Köpcke, Thor e Rahm (2010) e o número de novas classes de acordo com o conjunto de teste.

Tabela 5.11 – Métrica pF_1 obtida pelo método AssocIER, pelo *baseline* implementado e a apresentada por Köpcke, Thor e Rahm (2010) para as bases de dados *Abt-Buy*, *Amazon-Google*, *DBLP-ACM*, e *DBLP-Scholar*. A coluna novas classes apresenta o número de novas classes, de acordo com o conjunto de referência. Os melhores resultados estão destacados em negrito.

Bases de Dados	AssocIER	*Baseline*	Referência	Novas
	pF_1	pF_1	pF_1	Classes
Abt-Buy	**0.779**	0.597	0.713	63
Amazon-Google	**0.790**	0.524	0.601	76
DBLP-ACM	0.904	0.895	0.976	21
DBLP-Scholar	0.801	0.608	0.894	14

Fonte: Do autor (2017).

Na comparação com o *baseline* implementado, os resultados obtidos pelo método AssocIER são estatisticamente melhores em três das bases de dados e empata na base *DBLP-ACM*. No entanto, note que o resultado obtido com o *baseline* implementado neste trabalho é inferior aos apresentados por Köpcke, Thor e Rahm (2010), e nesse caso o método AssocIER é superior apenas para as bases de dados cujas entidades são produtos de lojas *online*. A grande maioria das

classes nessas bases de dados possui apenas dois registros, e nesse cenário o método AssocIER tende a não se comportar bem, principalmente porque o classificador associativo acaba composto por muitas regras de associação ruins. Ainda assim, pode-se considerar os resultados obtidos pelo método AssocIER satisfatórios.

Ressalta-se que o *baseline* implementado neste trabalho seguiu a descrição dada na referência, foi avaliado com as mesmas métricas e usou as configurações sugeridas, mas mesmo assim não foi possível gerar resultados mais próximos aos reportados pelos autores. As diferenças nos resultados podem ser atribuídas a diferenças nas execuções das experimentações, uma vez que alguns detalhes não foram apresentados pelos autores. Por exemplo, os autores não reportam os parâmetros utilizados para o treinamento do SVM, já que a melhor configuração é escolhida automaticamente pela ferramenta que os mesmos utilizaram em sua experimentação e à qual não tivemos acesso. Adicionalmente, é provável que a distribuição de registros entre os 10 *folds* não seja a mesma, dificultando ainda mais uma comparação direta e estatisticamente confiável dos resultados obtidos pelo *baseline* aqui implementado e os resultados apresentados em Köpcke, Thor e Rahm (2010).

5.9 Dificuldades, problemas e limitações

O objetivo desta seção é apresentar limitações do método proposto e alguns casos onde o método falha e discutir razões pelas quais as falhas ocorreram.

Algumas das limitações do método proposto são observadas no classificador associativo, originalmente proposto em Oliveira (2014) e Oliveira e Pereira (2017) e adaptado para permitir a sua atualização incremental neste trabalho. Uma possível fonte de falhas observada pelos autores diz respeito a *tokens* de baixo valor classificatório, mas que geram regras com 100% de confiança. Falhas desse tipo foram minimizadas com a utilização do suporte mínimo. Neste trabalho foi experimentada a técnica *coverage* para remoção de regras ruins (THABTAH, 2007). Essa técnica não só aumentou o tempo de execução do método como chegou a piorar os resultados em algumas das bases de dados testadas.

Outro cenário em que a predição pelo classificador associativo falha acontece quando duas classes ocorrem na fase de treinamento, e algum registro de uma das classes é subconjunto de algum registro da outra classe. É o que acontece por exemplo com as referências “Samsung Galaxy S7 Edge” e “Samsung Galaxy S7” de duas classes distintas. Os *tokens* na segunda descrição são um subconjunto da primeira, e assim nenhuma regra de associação é gerada para

a segunda classe. No classificador associativo original, se isso ocorre, e um registro da segunda classe ocorre no teste, tal registo não pode ser classificado com regras. Com o mecanismo de auto treinamento e detecção de novas classes propostas neste trabalho, se o registro da segunda classe ocorre no teste e nenhuma regra é encontrada, é provável que tal registro seja classificado como pertencente à primeira classe devido à alta similaridade, e esse erro será propagado com a atualização do módulo de aprendizagem.

Mais uma limitação do método AssocIER é a própria propagação de erros. Se algum registro de teste é classificado ou identificado incorretamente como pertencente a uma nova classe, esse erro é propagado com a atualização do modelo. Esse tipo de falha foi minimizada primariamente com o adiamento da classificação de novos registros caso a predição dos classificadores auxiliares não seja confiável. No classificador associativo, a inserção de uma nova regra de associação só ocorre se ainda não existe uma regra com o mesmo antecedente no modelo e desde que atenda tanto ao critério de confiança quanto ao suporte mínimo.

Outras limitações para o método podem ser originadas dos classificadores auxiliares. No caso do classificador baseado na similaridade do Cosseno, por exemplo, o tratamento de mudanças no conceito e evolução nos atributos é impraticável em termos de eficiência. A indexação de novos *tokens* e o recálculo do TF-IDF chegou a dobrar o tempo de execução em algumas bases de dados, mesmo quando realizado somente com registros classificados como pertencentes a novas classes. No *ensemble* utilizado nas experimentações, em que foi utilizado um classificador auxiliar baseado no Cosseno e outro baseado na similaridade de Jaccard, essa limitação explica os resultados inferiores quando não é utilizado nenhum dado para treinamento. Nesse caso, o módulo para detecção de novas classes possui informações somente do classificador baseado no Jaccard.

6 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, foi proposta e avaliada uma abordagem incremental para a resolução de entidades descritas por dados não estruturados. Uma abordagem incremental para essa tarefa é particularmente importante para aplicações Web que mantêm referências a milhares de entidades reais coletadas de diversas fontes, e onde novos dados estão constantemente disponíveis. Abordagens tradicionais para a resolução de entidades geralmente necessitam reavaliar toda a base de dados para agregar os novos dados à solução. A abordagem proposta neste trabalho provou-se efetiva e eficiente na resolução de entidades incrementalmente.

A abordagem proposta usa aprendizagem de máquina supervisionada e foi denominada AssocIER. Um classificador associativo e um ou mais classificadores auxiliares são usados para compor um *ensemble* de classificadores multi classes. AssocIER é capaz de detectar novas classes que emergem na fase de teste entre os novos dados e atualizar o modelo de aprendizagem incrementalmente com essas informações, expandindo a resolução de entidades. O mecanismo para detecção de novas classes junto ao auto treinamento do modelo de aprendizagem tratam o fenômeno evolução do conceito. Esses mecanismos e a utilização de uma estrutura de *hash* na implementação do classificador associativo ainda possibilitaram que a abordagem tratasse naturalmente a mudança de conceitos e a evolução de atributos.

AssocIER foi avaliada em várias bases de dados reais, compostas por registros de milhares de entidades diferentes, e a comparação com uma abordagem tradicional para resolução de entidades mostra que a abordagem proposta é mais escalável e mais efetiva que a abordagem tradicional na maioria das bases de dados testadas. Esses resultados atendem aos objetivos de uma abordagem incremental para resolução de entidades, conforme proposto por Gruenheid, Dong e Srivastava (2014). Isto é, AssocIER obteve resultados similares ou melhores à abordagem tradicional que executa em modo *batch* e o seu tempo de execução também é menor que o tempo de execução da abordagem tradicional.

A avaliação do mecanismo de auto treinamento em diferentes configurações do método AssocIER demonstrou a importância desse recurso, especialmente em bases de dados com poucos registros de dados por classe. A atualização do modelo de aprendizagem com novos registros de dados é importante, mesmo quando esses registros pertencem a classes já conhecidas pelo modelo de aprendizagem. Ademais, a atualização do modelo de aprendizagem é fundamental para o tratamento de mudanças nos conceitos e evolução dos atributos, através da criação

de novas regras de associação e possível remoção de regras que já pertencem ao classificador associativo.

O mecanismo para detecção de novas classes mostrou-se efetivo e capaz de identificar centenas de novas classes que não ocorreram em tempo de treinamento. A maioria dos erros na identificação de novas classes ocorrem devido à presença de regras de associação ruins no classificador associativo, e em último caso pela alta similaridade entre registros de classes diferentes. Algumas alternativas para minimizar o geração de regras ruins foram investigadas, como a eliminação de tais regras em tempo de treinamento e o pré-processamento adicional das descrições dos registros, mas nenhuma delas levou a ganhos.

A avaliação da abordagem AssocIER em cenários que poucos registros de dados estão disponíveis para a geração de um modelo inicial apresenta resultados satisfatórios. Essa avaliação ainda mostra que a abordagem pode ser executada sem nenhum dado para treinamento, caso em que o modelo de aprendizagem é totalmente gerado incrementalmente na fase de teste, e todas os registros são de novas classes. Mesmo com apenas 10% de dados do conjunto de treinamento, o método AssocIER ainda apresenta resultados superiores aos apresentados pelo *baseline* com os quais a abordagem proposta foi comparada.

Em trabalhos futuros, serão investigados em mais detalhes a aplicação da abordagem AssocIER no problema de desambiguação de nomes de autores, com a adição de um classificador que explore características específicas desse problema. Também será estudado um mecanismo para tratar melhor o fenômeno *concept-drift* (GAMA et al., 2014) em termos de espaço em memória, com o objetivo de permitir a remoção de registros de classes que não são usados há algum tempo. Isso pode ser facilmente implementado no classificador associativo do AssocIER. Por fim, será implementada uma versão distribuída do método para execução em ambiente Apache Spark (KARAU et al., 2015), com o objetivo de fazer AssocIER ser capaz de lidar com grandes volumes de dados.

REFERÊNCIAS

AGRAWAL, R.; IMIELINSKI, T.; SWAMI, A. Mining association rules between sets of items in large databases. **ACM SIGMOD Record**, ACM, New York, NY, USA, v. 22, n. 2, p. 207–216, jun 1993. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/170036.170072>.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules in large databases. In: **Proceedings of the 20th International Conference on Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994. (VLDB '94), p. 487–499. ISBN 1-55860-153-8. Disponível em: <http://dl.acm.org/citation.cfm?id=645920.672836>.

ALBERTINI, M. K.; MELLO, R. F. de. A self-organizing neural network for detecting novelties. In: **Proceedings of the 2007 ACM Symposium on Applied Computing**. New York, NY, USA: ACM, 2007. (SAC '07), p. 462–466. ISBN 1-59593-480-4. Disponível em: <http://doi.acm.org/10.1145/1244002.1244110>.

ALTOWIM, Y.; KALASHNIKOV, D. V.; MEHROTRA, S. Progressive approach to relational entity resolution. **Proc. VLDB Endow.**, VLDB Endowment, v. 7, n. 11, p. 999–1010, jul. 2014. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2732967.2732975>.

ALY, M. Survey on multiclass classification methods. **Technical Report, Caltech**, California Institute of Technology, 2005.

ANTONIE, M. L.; ZAÏANE, O. R. Text document categorization by term association. In: **ICDM: Proceedings of the 2002 IEEE International Conference on Data Mining**. Washington, DC, USA: IEEE Computer Society, 2002. p. 19+. ISBN 0-7695-1754-4. Disponível em: <http://portal.acm.org/citation.cfm?id=844745>.

BAEZA-YATES, R.; RIBEIRO-NETO, B. **Modern information retrieval: The Concepts and Technology behind Search**. Wokingham, UK: Addison-Wesley Professional, 2011.

BALAJI, J.; JAVED, F.; KEJRIWAL, M.; MIN, C.; SANDER, S.; OZTURK, O. An ensemble blocking scheme for entity resolution of large and sparse datasets. **CoRR**, abs/1609.06265, 2016. Disponível em: <http://arxiv.org/abs/1609.06265>.

BALAKRISHNAN, V.; LLOYD-YEMOH, E. Stemming and lemmatization: a comparison of retrieval performances. **Lecture Notes on Software Engineering**, IACSIT Press, v. 2, n. 3, p. 262, 2014.

BANDA, R. **Indexing techniques for real-time entity resolution**. Tese (Doutorado) — College of Engineering and Computer Science, Australian National University, Camberra, Australia, Mar 2016.

BAXTER, R.; CHRISTEN, P. A comparison of fast blocking methods for record linkage. In: **Proceedings of the KDD Workshop Data Cleaning, Record Linkage, and Object Consolidation**. Washington, DC: ACM, 2003. p. 25–27.

BENJELLOUN, O.; GARCIA-MOLINA, H.; MENESTRINA, D.; SU, Q.; WHANG, S. E.; WIDOM, J. Swoosh: A generic approach to entity resolution. **The VLDB Journal**, Springer-Verlag New York, v. 18, n. 1, p. 255–276, January 2009. ISSN 1556-4681. Disponível em: <http://dx.doi.org/10.1007/s00778-008-0098-x>.

BERRY, M. J. A.; LINOFF, G. **Data Mining Techniques: for Marketing, Sales and Customer Support**. New York, NY, USA: Wiley Computer Publishing, 1997.

BHATTACHARYA, I.; GETOOR, L. Iterative record linkage for cleaning and integration. In: **Proceedings of the 9th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery**. New York, NY, USA: ACM, 2004. (DMKD '04), p. 11–18. ISBN 1-58113-908-X. Disponível em: <http://doi.acm.org/10.1145/1008694.1008697>.

BHATTACHARYA, I.; GETOOR, L. Collective entity resolution in relational data. **ACM Transaction on Knowledge Discovery from Data**, ACM, New York, NY, USA, v. 1, n. 1, p. 5, 2007. ISSN 1556-4681.

BHATTACHARYA, I.; GETOOR, L. Online collective entity resolution. In: **Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2**. AAAI Press, 2007. (AAAI'07), p. 1606–1609. ISBN 978-1-57735-323-2. Disponível em: <http://dl.acm.org/citation.cfm?id=1619797.1619903>.

BHATTACHARYA, I.; GETOOR, L.; LICAMELE, L. Query-time entity resolution. In: **Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2006. (KDD '06), p. 529–534. ISBN 1-59593-339-5. Disponível em: <http://doi.acm.org/10.1145/1150402.1150463>.

BILENKO, M.; BASIL, S.; SAHAMI, M. Adaptive product normalization: using online learning for record linkage in comparison shopping. In: **Proceedings of the Fifth IEEE International Conference on Data Mining**. Washington, DC, USA: IEEE Computer Society, 2005. p. 58–65. ISSN 1550-4786.

BILENKO, M.; MOONEY, R. Adaptive duplicate detection using learnable string similarity measures. In: **Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. Washington, USA: ACM, New York, NY, USA, 2003. p. 39–48. ISBN 1-58113-737-0.

BILENKO, M.; MOONEY, R.; COHEN, W.; RAVIKUMAR, P.; FIENBERG, S. Adaptive name matching in information integration. **IEEE Intelligent Systems**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 18, n. 5, p. 16–23, 2003. ISSN 1541-1672.

BOUZOUITA, I.; ELLOUMI, S. Generic associative classification rules: A comparative study. **International Journal of Advanced Science and Technology,(33)**, p. 69–84, 2011.

BREIMAN, L. Bagging predictors. **Machine Learning**, Kluwer Academic Publishers-Plenum Publishers, v. 24, n. 2, p. 123–140, 1996. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A%3A1018054314350>.

BREIMAN, L. Random forests. **Machine Learning**, Springer Netherlands, v. 45, n. 1, p. 5–32, October 2001. ISSN 0885-6125.

CARVALHO, M. G. de; GONÇALVES, M. A.; LAENDER, A. H. F.; SILVA, A. S. da. Learning to deduplicate. In: **Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries**. New York, NY, USA: ACM, 2006. p. 41–50. ISBN 1-59593-354-9. Disponível em: <http://doi.acm.org/10.1145/1141753.1141760>.

CHANG, W.; CHENG, J.; ALLAIRE, J.; XIE, Y.; MCPHERSON, J. Computer Program, **shiny: Web Application Framework for R. R package version 0.12.1**. 2015. Disponível em: <http://CRAN.R-project.org/package=shiny>.

CHAUDHURI, S.; GANTI, V.; MOTWANI, R. Robust identification of fuzzy duplicates. In: **Proceedings of the 21st International Conference on Data Engineering**. Washington, DC, USA: IEEE Computer Society, 2005. (ICDE '05), p. 865–876. ISBN 0-7695-2285-8. Disponível em: <http://dx.doi.org/10.1109/ICDE.2005.125>.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine Learning**, Kluwer Academic Publishers, Hingham, MA, USA, v. 20, n. 3, p. 273–297, sep 1995. ISSN 0885-6125. Disponível em: <http://dx.doi.org/10.1023/A:1022627411411>.

COSTA, G.; MANCO, G.; ORTALE, R. An incremental clustering scheme for data de-duplication. **Data Min. Knowl. Discov.**, Kluwer Academic Publishers, Hingham, MA, USA, v. 20, n. 1, p. 152–187, jan. 2010. ISSN 1384-5810. Disponível em: <http://dx.doi.org/10.1007/s10618-009-0155-0>.

DALLACHIESA, M.; EBAID, A.; ELDAWY, A.; ELMAGARMID, A.; ILYAS, I. F.; OUZZANI, M.; TANG, N. Nadeef: A commodity data cleaning system. In: **Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: ACM, 2013. (SIGMOD '13), p. 541–552. ISBN 978-1-4503-2037-5. Disponível em: <http://doi.acm.org/10.1145/2463676.2465327>.

DEY, D.; MOOKERJEE, V.; LIU, D. Efficient techniques for online record linkage. **IEEE Trans. on Knowl. and Data Eng.**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 23, n. 3, p. 373–387, mar. 2011. ISSN 1041-4347. Disponível em: <http://dx.doi.org/10.1109/TKDE.2010.134>.

EFTHYMIOU, V.; PAPADAKIS, G.; PAPASTEFANATOS, G.; STEFANIDIS, K.; PALPANAS, T. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. **Inf. Syst.**, v. 65, p. 137–157, 2017. Disponível em: <http://dx.doi.org/10.1016/j.is.2016.12.001>.

EFTHYMIOU, V.; STEFANIDIS, K.; CHRISTOPHIDES, V. Minoan ER: progressive entity resolution in the web of data. In: **Proceedings of the 19th International Conference on Extending Database Technology, EDBT 2016, Bordeaux, France, March 15-16, 2016, Bordeaux, France, March 15-16, 2016.** OpenProceedings.org, 2016. p. 670–671. Disponível em: <http://dx.doi.org/10.5441/002/edbt.2016.79>.

ELMACIOGLU, E.; KAN, M.-Y.; LEE, D.; ZHANG, Y. Web based linkage. In: **Proceedings of the 9th Annual ACM International Workshop on Web Information and Data Management**. Lisbon, Portugal: ACM, 2007. p. 121–128. ISBN 978-1-59593-829-9.

ELMAGARMID, A. K.; IPEIROTIS, P. G.; VERYKIOS, V. S. Duplicate record detection: A survey. **IEEE Transaction on Knowledge and Data Engineering**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 19, n. 1, p. 1–16, 2007. ISSN 1041-4347.

ENRíQUEZ, J.; DOMíNGUEZ-MAYO, F.; ESCALONA, M.; ROSS, M.; STAPLES, G. Entity reconciliation in big data sources: a systematic mapping study. **Expert Systems with Applications**, p. –, 2017. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417417301550>.

FARIA, E. R. de; CARVALHO, A. C. Ponce de L. F.; GAMA, J. Minas: multiclass learning algorithm for novelty detection in data streams. **Data Mining and Knowledge Discovery**, v. 30, n. 3, p. 640–680, 2016. ISSN 1573-756X. Disponível em: <http://dx.doi.org/10.1007/s10618-015-0433-y>.

FELLEGI, I. P.; SUNTER, A. B. A theory for record linkage. **Journal of the American Statistical Association**, v. 64, n. 328, p. 1183–1210, 1969.

FERREIRA, A. A.; GONÇALVES, M. A.; LAENDER, A. H. F. A brief survey of automatic methods for author name disambiguation. **SIGMOD Record**, v. 41, n. 2, p. 15–26, 2012.

FERREIRA, A. A.; VELOSO, A.; GONÇALVES, M. A.; LAENDER, A. H. Effective self-training author name disambiguation in scholarly digital libraries. In: **Proceedings of the 10th Annual Joint Conference on Digital Libraries**. New York, NY, USA: ACM, 2010. (JCDL '10), p. 39–48. ISBN 978-1-4503-0085-8. Disponível em: <http://doi.acm.org/10.1145/1816123.1816130>.

FERREIRA, A. A.; VELOSO, A.; GONÇALVES, M. A.; LAENDER, A. H. Self-training author name disambiguation for information scarce scenarios. **Journal of the Association for Information Science and Technology**, Wiley Online Library, v. 65, n. 6, p. 1257–1278, 2014.

FRENCH, J. C.; POWELL, A. L.; SCHULMAN, E. Using clustering strategies for creating authority files. **Journal of the American Society for Information Science**, v. 51, n. 8, p. 774–786, 2000.

FREUND, Y.; SCHAPIRE, R. E. Experiments with a new boosting algorithm. In: SAITTA, L. (Ed.). **Proceedings of the Thirteenth International Conference on Machine Learning (ICML 1996)**. Morgan Kaufmann, 1996. p. 148–156. ISBN 1-55860-419-7. Disponível em: <http://www.biostat.wisc.edu/~kbroman/teaching/statgen/2004/refs/freund.pdf>.

GALHARDAS, H.; FLORESCU, D.; SHASHA, D.; SIMON, E.; SAITA, C.-A. Declarative data cleaning: Language, model, and algorithms. In: **Proceedings of the 27th International Conference on Very Large Data Bases**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001. (VLDB '01), p. 371–380. ISBN 1-55860-804-4. Disponível em: <http://dl.acm.org/citation.cfm?id=645927.672042>.

GAMA, J.; ZLIOBAITĖ, I.; BIFET, A.; PECHENIZKIY, M.; BOUCHACHIA, A. A survey on concept drift adaptation. **ACM Computing Surveys**, ACM, New York, NY, USA, v. 46, n. 4, p. 44:1–44:37, 2014. ISSN 0360-0300. Disponível em: <http://doi.acm.org/10.1145/2523813>.

GETOOR, L.; MACHANAVAJJHALA, A. Entity resolution: Theory, practice and open challenges. **Proceedings of the VLDB Endowment**, v. 5, n. 12, p. 2018–2019, 2012. Tutorial available at http://www.cs.umd.edu/∼getoor/Tutorials/ER_VLDB2012.pdf.

GRUENHEID, A.; DONG, X. L.; SRIVASTAVA, D. Incremental record linkage. **Proceedings of the VLDB Endowment**, VLDB Endowment, v. 7, n. 9, p. 697–708, May 2014. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2732939.2732943>.

HALL, M.; FRANK, E.; HOLMES, G.; PFAHRINGER, B.; REUTEMANN, P.; WITTEN, I. H. The weka data mining software: An update. **SIGKDD Explor. Newsl.**, ACM, New York, NY, USA, v. 11, n. 1, p. 10–18, nov 2009. ISSN 1931-0145. Disponível em: <http://doi.acm.org/10.1145/1656274.1656278>.

HAN, H.; GILES, L.; ZHA, H.; LI, C.; TSIOUTSIOULIKLIS, K. Two supervised learning approaches for name disambiguation in author citations. In: **Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries**. New York, NY, USA: ACM, 2004. (JCDL '04), p. 296–305. ISBN 1-58113-832-6. Disponível em: <http://doi.acm.org/10.1145/996350.996419>.

HERNÁNDEZ, M. A.; STOLFO, S. J. The merge/purge problem for large databases. **ACM SIGMOD Record**, ACM, New York, NY, USA, v. 24, n. 2, p. 127–138, may 1995. ISSN 0163-5808. Disponível em: <http://doi.acm.org/10.1145/568271.223807>.

HERZOG, T. N.; SCHEUREN, F. J.; WINKLER, W. E. **Data Quality and Record Linkage Techniques**. 1st. ed. New York, NY, USA: Springer Publishing Company, Incorporated, 2007. ISBN 0387695028, 9780387695020.

JACCARD, P. étude comparative de la distribuition florale dans une portion des alpes et des jura. **Bulletin del la Société Vaudoise des Sciences Naturelles**, v. 37, p. 547–579, 1901.

JARO, M. A. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. **Journal of the American Statistical Association**, v. 84, n. 406, p. 414–420, June 1989.

JONES, K. S. A statistical interpretation of term specificity and its application in retrieval. **Journal of Documentation**, v. 28, p. 11–21, 1972.

JUNG, C. F. **Metodologia para pesquisa e desenvolvimento: aplicada a novas tecnologias, produtos e processos**. Rio de Janeiro, RJ, Brasil: Axcel Books, 2004.

KALASHNIKOV, D. V.; MEHROTRA, S. Domain-independent data cleaning via analysis of entity-relationship graph. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 31, n. 2, p. 716–767, jun. 2006. ISSN 0362-5915. Disponível em: <http://doi.acm.org/10.1145/1138394.1138401>.

KARAU, H.; KONWINSKI, A.; WENDELL, P.; ZAHARIA, M. **Learning Spark**. Sebastopol, CA, USA: O'Reilly, 2015. ISBN 978-1-449-35862-4.

KOLB, L.; KöPCKE, H.; THOR, A.; RAHM, E. Learning-based entity resolution with mapreduce. In: **Proceedings of the third international workshop on Cloud data management**. ACM, New York, NY, USA, 2011. p. 1–6. ISBN 978-1-4503-0956-1. Disponível em: <http://doi.acm.org/10.1145/2064085.2064087>.

KONDRAK, G. N-gram similarity and distance. In: **Proceedings of the 12th International Conference on String Processing and Information Retrieval**. Berlin, Heidelberg: Springer-Verlag, 2005. (SPIRE'05), p. 115–126. ISBN 3-540-29740-5, 978-3-540-29740-6. Disponível em: <http://dx.doi.org/10.1007/11575832_13>.

KÖPCKE, H.; RAHM, E. Training selection for tuning entity matching. In: **Proceedings of the International Workshop on Quality in Databases and Management of Uncertain Data, Auckland, New Zealand, August 2008**. Enschede, The Netherlands: Centre for Telematics and Information Technology, University of Twente, 2008. p. 3–12. Disponível em: <http://www.vldb.org/conf/2008/workshops/WProc_qdbmud/linkage1.pdf>.

KÖPCKE, H.; RAHM, E. Frameworks for entity matching: A comparison. **Data & Knowledge Engineering**, v. 69, n. 2, p. 197–210, 2010. ISSN 0169-023X. Disponível em: <http://dx.doi.org/10.1016/j.datak.2009.10.003>.

KÖPCKE, H.; THOR, A.; RAHM, E. Evaluation of entity resolution approaches on real-world match problems. **Proceedings of the VLDB Endowment**, v. 3, n. 1–2, p. 484–493, 2010. ISSN 2150-8097. Disponível em: <http://dl.acm.org/citation.cfm?id=1920841.1920904>.

KÖPCKE, H.; THOR, A.; THOMAS, S.; RAHM, E. Tailoring entity resolution for matching product offers. In: **Proceedings of the 15th International Conference on Extending Database Technology**. Berlin, Germany: ACM, New York, NY, USA, 2012. p. 545–550. ISBN 978-1-4503-0790-1. Disponível em: <http://doi.acm.org/10.1145/2247596.2247662>.

KOUDAS, N.; SARAWAGI, S.; SRIVASTAVA, D. Record linkage: Similarity measures and algorithms. In: **Proceedings of the ACM SIGMOD International Conference on Management of Data**. Chicago, USA: ACM New York, NY, USA, 2006. p. 802–803. ISBN 1-59593-434-0.

LAPIDOT, I. **Self-Organizing-Maps with BIC for Speaker Clustering**. Martigny, Switzerland, 2002.

LEE, D.; KANG, J.; MITRA, P.; GILES, C. L.; ON, B.-W. Are your citations clean? **Communications of the ACM**, v. 50, n. 12, p. 33–38, December 2007.

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions, and reversals. **Soviet Physics Doklady**, v. 10, n. 8, p. 707–710, 1966.

LI, W.; HAN, J.; PEI, J. Cmar: Accurate and efficient classification based on multiple class-association rules. In: IEEE. **Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on**. 2001. p. 369–376. Disponível em: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=989541>.

LibSVM Package. 2009.
www.csie.ntu.edu.tw/∼cjlin/libsvm. Accessed in January, 2009.

MALHOTRA, P.; AGARWAL, P.; SHROFF, G. Graph-parallel entity resolution using LSH & IMM. In: **Proceedings of the Workshops of the EDBT/ICDT 2014 Joint Conference (EDBT/ICDT 2014), Athens, Greece, March 28, 2014.** CEUR-WS.org, 2014. p. 41–49. Disponível em: <http://ceur-ws.org/Vol-1133/paper-07.pdf>.

MALHOTRA, P.; AGARWAL, P.; SHROFF, G. Incremental entity fusion from linked documents. In: **17th International Conference on Information Fusion (FUSION)**. Salamanca, Salamanca, Spain: IEEE Explore, 2014. p. 1–8.

MANNING, C. D.; SURDEANU, M.; BAUER, J.; FINKEL, J.; BETHARD, S. J.; MCCLOSKY, D. The Stanford CoreNLP natural language processing toolkit. In: **Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations**. Baltimore, USA: Association for Computational Linguistics, 2014. p. 55–60.

MASUD, M.; CHEN, Q.; KHAN, L.; AGGARWAL, C.; GAO, J.; HAN, J.; SRIVASTAVA, A.; OZA, N. Classification and adaptive novel class detection of feature-evolving data streams. **IEEE Transactions on Knowledge and Data Engineering**, v. 25, n. 7, p. 1484–1497, July 2013. ISSN 1041-4347.

MCCALLUM, A.; NIGAM, K.; UNGAR, L. H. Efficient clustering of high-dimensional data sets with application to reference matching. In: **Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. Boston, USA: ACM, New York, NY, USA, 2000. p. 169–178. ISBN 1-58113-233-6.

MCCALLUM, A. K.; NIGAM, K.; RENNIE, J.; SEYMORE, K. Automating the construction of internet portals with machine learning. **Information Retrieval**, v. 3, n. 2, p. 127–163, 2000. ISSN 1573-7659. Disponível em: <http://dx.doi.org/10.1023/A:1009953814988>.

MITCHELL, T. M. **Machine Learning**. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MONGE, A. E.; ELKAN, C. P. The field matching problem: Algorithms and applications. In: **Proceedings of the Second International Conference on Knowledge Discovery and Data Mining**. AAAI Press, 1996. (KDD'96), p. 267–270. Disponível em: <http://dl.acm.org/citation.cfm?id=3001460.3001516>.

NEWCOMBE, H. B.; KENNEDY, J. M.; AXFORD, S. J.; JAMES, A. P. Automatic linkage of vital records. **Science**, v. 130, n. 3381, p. 954–959, October 1959. ISSN 0036-8075.

NURAY-TURAN, R.; KALASHNIKOV, D. V.; MEHROTRA, S. Exploiting web querying for web people search. **ACM Trans. Database Syst.**, ACM, New York, NY, USA, v. 37, n. 1, p. 7:1–7:41, mar. 2012. ISSN 0362-5915. Disponível em: <http://doi.acm.org/10.1145/2109196.2109203>.

OLIVEIRA, C. M. **Um método baseado em regras de associação para classificação de ofertas de produtos de lojas de comércio eletrônico**. Dissertação (Mestrado) — Universidade Federal de Lavras, Minas Gerais - Brasil, 2014.

OLIVEIRA, C. M.; PEREIRA, D. A. An association rules based method for classifying product offers from e-shopping. **Intelligent Data Analysis**, v. 21, n. 3, 2017. Accepted for publication.

PAIVA, E. R. d. F. **Detecção de novidade em fluxos contínuos de dados multiclasse**. Tese (Doutorado) — Universidade de São Paulo, 2014.

PAPADAKIS, G.; PAPASTEFANATOS, G.; KOUTRIKA, G. Supervised meta-blocking. **Proc. VLDB Endow.**, VLDB Endowment, v. 7, n. 14, p. 1929–1940, out. 2014. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2733085.2733098>.

PAPENBROCK, T.; HEISE, A.; NAUMANN, F. Progressive duplicate detection. **IEEE Transactions on Knowledge and Data Engineering**, v. 27, n. 5, p. 1316–1329, May 2015. ISSN 1041-4347.

PASULA, H.; MARTHI, B.; MILCH, B.; RUSSELL, S.; SHPITSER, I. Identity uncertainty and citation matching. In: **Proceedings of the Advances in Neural Information Processing Systems**. Vancouver, Canada: MIT Press, 2002. p. 1401–1408.

PEREIRA, D. A.

A Web-based Approach for Entity Resolution and Creation of Authority Files, December 2009. PhD Thesis, Department of Computer Science, Federal University of Minas Gerais, Brazil.

PEREIRA, D. A.; RIBEIRO-NETO, B.; ZIVIANI, N.; LAENDER, A. H. F.; GONÇALVES, M. A. A generic web-based entity resolution framework. **Journal of the American Society for Information Science and Technology**, v. 62, n. 5, p. 919–932, May 2011. ISSN 1532-2890.

PEREIRA, D. A.; SILVA, E. E. B. da; ESMIN, A. A. A. Disambiguating publication venue titles using association rules. In: **IEEE/ACM Joint Conference on Digital Libraries**. London, UK: IEEE Press, 2014. p. 77–86.

PERSIN, M. Document filtering for fast ranking. In: **Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval**. New York, NY, USA: Springer-Verlag New York, Inc., 1994. (SIGIR'94), p. 339–348. ISBN 0-387-19889-X. Disponível em: <http://dl.acm.org/citation.cfm?id=188490.188597>.

POLIKAR, R. Ensemble based systems in decision making. **Circuits and Systems Magazine, IEEE**, v. 6, n. 3, p. 21–45, Third 2006. ISSN 1531-636X.

PRODANOV, C. C.; FREITAS, E. C. de. **Metodologia do Trabalho Científico: Métodos e Técnicas da Pesquisa e do Trabalho Acadêmico-2ª Edição**. Novo Hamburgo, RS, Brasil: Editora Feevale, 2013. ISBN 978-85-7717-158-3.

R Core Team. R: A language and environment for statistical computing. 2015. Disponível em: <http://www.R-project.org>.

RAMADAN, B.; CHRISTEN, P.; LIANG, H.; GAYLER, R. W. Dynamic sorted neighborhood indexing for real-time entity resolution. **J. Data and Information Quality**, ACM, New York, NY, USA, v. 6, n. 4, p. 15:1–15:29, out. 2015. ISSN 1936-1955. Disponível em: <http://doi.acm.org/10.1145/2816821>.

ROKACH, L. Ensemble-based classifiers. **Artificial Intelligence Review**, Springer Netherlands, v. 33, n. 1-2, p. 1–39, 2010. ISSN 0269-2821. Disponível em: <http://dx.doi.org/10.1007/s10462-009-9124-7>.

SALTON, G.; MCGILL, M. J. **Introduction to Modern Information Retrieval**. New York, NY, USA: McGraw-Hill, Inc., 1983. ISBN 0070544840.

SANTANA, A. F.; GONÇALVES, M. A.; LAENDER, A. H. F.; FERREIRA, A. A. On the combination of domain-specific heuristics for author name disambiguation: the nearest cluster method. **International Journal on Digital Libraries**, v. 16, n. 3, p. 229–246, 2015. ISSN 1432-1300. Disponível em: <http://dx.doi.org/10.1007/s00799-015-0158-y>.

SARMA, A. D.; JAIN, A.; MACHANAVAJJHALA, A.; BOHANNON, P. An automatic blocking mechanism for large-scale de-duplication tasks. In: **Proceedings of the 21st ACM International Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2012. (CIKM '12), p. 1055–1064. ISBN 978-1-4503-1156-4. Disponível em: <http://doi.acm.org/10.1145/2396761.2398403>.

SØRENSEN, T. A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. **Biologiske Skrifter**, v. 5, p. 1–34, 1948.

SPINOSA, E. J.; CARVALHO, A. P. de Leon F. de; GAMA, J. Novelty detection with application to data streams. **Intelligent Data Analysis**, IOS Press, Amsterdam, The Netherlands, The Netherlands, v. 13, n. 3, p. 405–422, aug 2009. ISSN 1088-467X. Disponível em: <http://dl.acm.org/citation.cfm?id=1551768.1551770>.

STONEBRAKER, M.; BESKALES, G.; PAGAN, A.; BRUCKNER, D.; CHERNIACK, M.; XU, S.; ANALYTICS, V.; ILYAS, I. F.; ZDONIK, S. Data curation at scale: The data tamer system. In: **CIDR 2013**. Monterey, Canada: www.cidrdb.org, 2013.

TALBURT, J. R. **Entity Resolution and Information Quality**. 1st. ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010. ISBN 0123819725, 9780123819727.

TAN, P.-N.; STEINBACH, M.; KUMAR, V. **Introduction to Data Mining - Mineração de Dados**. Rio de Janeiro, RJ, Brasil: Editora Ciência Moderna Ltda, 2009. ISBN 9788573937619.

THABTAH, F. A review of associative classification mining. **The Knowledge Engineering Review**, Cambridge Univ Press, v. 22, n. 01, p. 37–65, 2007.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. New York, USA: Springer-Velag, 1995. ISBN 0-387-94559-8.

VELOSO, A.; FERREIRA, A. A.; GONÇALVES, M. A.; LAENDER, A. H. F.; JR., W. M. Cost-effective on-demand associative author name disambiguation. **Information Processing and Management**, v. 48, n. 4, p. 680–697, 2012.

VESDAPUNT, N.; BELLARE, K.; DALVI, N. Crowdsourcing algorithms for entity resolution. **Proc. VLDB Endow.**, VLDB Endowment, v. 7, n. 12, p. 1071–1082, ago. 2014. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2732977.2732982>.

WANG, J.; LI, G.; KRASKA, T.; FRANKLIN, M. J.; FENG, J. Leveraging transitive relations for crowdsourced joins. In: **Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data**. New York, NY, USA: ACM, 2013. (SIGMOD '13), p. 229–240. ISBN 978-1-4503-2037-5. Disponível em: <http://doi.acm.org/10.1145/2463676.2465280>.

WANG, X.; YUE, K.; NIU, W.; SHI, Z. An approach for adaptive associative classification. **Expert Systems with Applications**, Elsevier, v. 38, n. 9, p. 11873–11883, 2011.

WEDYAN, S. Review and comparison of associative classification data mining approaches. **International Journal of Computer, Electrical, Automation, Control and Information Engineering**, World Academy of Science, Engineering and Technology, v. 8, n. 1, p. 34–45, 2014. ISSN 1307-6892. Disponível em: <http://waset.org/publications/9997152>.

WELCH, M. J.; SANE, A.; DROME, C. Fast and accurate incremental entity resolution relative to an entity knowledge base. In: **Proceedings of the 21st ACM International Conference on Information and Knowledge Management**. New York, NY, USA: ACM, 2012. (CIKM'12), p. 2667–2670. ISBN 978-1-4503-1156-4. Disponível em: <http://doi.acm.org/10.1145/2396761.2398719>.

WHANG, S. E.; GARCIA-MOLINA, H. Incremental entity resolution on rules and data. **The VLDB Journal**, Springer-Verlag New York, Inc., Secaucus, NJ, USA, v. 23, n. 1, p. 77–102, February 2014. ISSN 1066-8888. Disponível em: <http://dx.doi.org/10.1007/s00778-013-0315-0>.

WHANG, S. E.; LOFGREN, P.; GARCIA-MOLINA, H. Question selection for crowd entity resolution. **Proc. VLDB Endow.**, VLDB Endowment, v. 6, n. 6, p. 349–360, abr. 2013. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/2536336.2536337>.

WHANG, S. E.; MARMAROS, D.; GARCIA-MOLINA, H. Pay-as-you-go entity resolution. **IEEE Trans. on Knowl. and Data Eng.**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 25, n. 5, p. 1111–1124, maio 2013. ISSN 1041-4347. Disponível em: <http://dx.doi.org/10.1109/TKDE.2012.43>.

WINKLER, W. E. The state of record linkage and current research problems. **Statistical Research Division, U.S. Census Bureau**, 1999. Internal Revenue Service Publication R99/04, http://www.census.gov/srd/papers/pdf/rr99-04.pdf.

WINKLER, W. E. Approximate string comparator search strategies for very large administrative lists. **Statistical Research Division**, Citeseer, Washington, DC, USA, 2005.

WOLPERT, D. H. Stacked generalization. **Neural networks**, Elsevier, v. 5, n. 2, p. 241–259, 1992.

YAKOUT, M.; ELMAGARMID, A. K.; ELMELEEGY, H.; OUZZANI, M.; QI, A. Behavior based record linkage. **Proc. VLDB Endow.**, VLDB Endowment, v. 3, n. 1-2, p. 439–448, set. 2010. ISSN 2150-8097. Disponível em: <http://dx.doi.org/10.14778/1920841.1920899>.

YI, L.; XING-CHUN, D.; JIAN-JUN, C.; XING, Z.; YU-LING, S. A method for entity resolution in high dimensional data using ensemble classifiers. **Mathematical Problems in Engineering**, Hindawi Publishing Corporation, v. 2017, 2017.

YIN, X.; HAN, J. Cpar: Classification based on predictive association rules. In: SIAM. **Proceedings of the third SIAM international conference on data mining**. San Francisco, CA, USA, 2003. v. 3, p. 369–376.

ZADEH, L. Fuzzy sets. **Information and Control**, v. 8, n. 3, p. 338 – 353, 1965. ISSN 0019-9958. Disponível em: <http://www.sciencedirect.com/science/article/pii/S001999586590241X>.

ZHANG, Y.; MUKHERJEE, R.; SOETARMAN, B. Concept extraction and e-commerce applications. **Electronic Commerce Research and Applications**, v. 12, n. 4, p. 289–296, 2013. ISSN 1567-4223. Social Commerce- Part 2. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1567422313000227>.