UNIVERSIDADE FEDERAL DO AMAZONAS - UFAM

INSTITUTO DE COMPUTAÇÃO - ICOMP

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA - PPGI

DIEGO DE AZEVEDO RODRIGUES

CASAMENTO DE ESQUEMAS DE BANCO DE DADOS APLICANDO APRENDIZADO ATIVO

Manaus, AM

2013

DIEGO DE AZEVEDO RODRIGUES

CASAMENTO DE ESQUEMAS DE BANCO DE DADOS APLICANDO APRENDIZADO ATIVO

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Amazonas - UFAM como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Prof. D.Sc. Altigran Soares da Silva

Manaus, AM

2013

Resumo

Dados dois esquemas de bancos de dados pertencentes ao mesmo domínio, o problema de Casamento de Esquemas consiste em encontrar pares de elementos desses esquemas que possuam a mesma semântica para aquele domínio. Tradicionalmente, tal tarefa era realizada manualmente por um especialista, tornando-a custosa e cansativa pois, este deveria conhecer bem os esquemas e o domínio em que estes estavam inseridos. Atualmente, esse processo é assistido por métodos semi-automáticos de casamento de esquemas. Os métodos atuais utilizam diversas heurísticas para gerar os casamentos e muitos deles compartilham uma modelagem em comum: constroem uma matriz de similaridade entre os elementos a partir de funções chamadas *matchers* e, baseados nos valores dessa matriz, decidem segundo algum critério quais os casamentos válidos. Esta dissertação apresenta um método baseado em aprendizado ativo que utiliza a matriz de similaridade gerada pelos *matchers* e um algoritmo de aprendizagem de máquina, além de intervenções de um especialista, para gerar os casamentos. O método apresentado se diferencia dos outros por não possuir uma heurística fixa e por utilizar a experiência do especialista apenas quando necessário. Em nossos experimentos, avaliamos o método proposto contra um *baseline* em dois *datasets*: o primeiro que foi o mesmo utilizado pelo *baseline* e o segundo contendo esquemas propostos em um *benchmark* para integração de esquemas. Mostramos que o *baseline* alcança bons resultados no *dataset* em que foi originalmente testado, mas que sua estratégia fixa não é tão efetiva para outros esquemas. Por outro lado, o método baseado em aprendizado ativo que propomos se mostra consistente em ambos os *datasets*, alcançando, em média, um valor de medida-F igual a $0,64$.

Palavras-chave: casamento de esquemas, integração de dados, aprendizado ativo.

Abstract

Given two database schemas within the same domain, the schema matching problem is the task of finding pairs of schema elements that have the same semantics for that domain. Usually, this task was performed manually by a specialist making it tedious and costly because the specialist should know the schemas and their domain. Currently this process is assisted by semi-automatic schema matching methods. Current, methods use some heuristics to generate matchings and many of them share a common modeling: they build a similarity matrix between the elements from functions called matchers and, based on the matrix values, decide according to a criterion which of the matchings are correct. This thesis presents an active-learning based method that uses the similarity matrix generated by the matchers, a machine learning algorithm and specialist interventions to generate matchings. The presented method differs from others because it has no fixed heuristic and uses the specialist expertise only when necessary. In our experiments, we evaluate the proposed method against a baseline on two datasets: the first one was the same used by the baseline and the second containing schemas of a benchmark for schema integration. We show that baseline achieves good results on its original dataset, but its fixed strategy is not as effective for other schemas. Moreover, the proposed method based on active learning is shown more consistent achieving, on average, F-measure value of 0.64.

Keywords: schema matching, data integration, active learning.

Agradecimentos

Agradeço a Deus, por iluminar meu caminho e me dar forças durante essa caminhada.

À toda a minha família, em especial à minha mãe Helena, meu padrasto Dalmir e meus irmãos Athos e Darla. Obrigado pela minha educação e pelo apoio a mim concedido.

Ao meu orientador, Prof. Dr. Altigran Soares, pela oportunidade e confiança depositadas, pelas horas dedicadas e pelos conselhos dados.

À minha co-orientadora, Profª. Drª. Rosiane de Freitas, pela ajuda, sobretudo no início deste trabalho, e pelo apoio durante o processo.

Aos membros da banca, Profª. Drª. Eulanda dos Santos e Profª. Drª. Bernadette Lóscio, pelas críticas e sugestões fornecidas para o enriquecimento deste trabalho.

À minha prima Rannah, pela amizade e pelas conversas (e por tantas merendas preparadas!). À tia Graça, ao tio José Moraes e às minhas primas (carinhosamente chamadas de tias) Ítala e Keylah, pelos momentos de descontração e alegria.

À minha amiga Anna, pelos anos de amizade, pelas bagunças, pelos conselhos.

Aos meus amigos Jéssica e Ramisson, pela amizade de longa data e pelo companheirismo durante todos esses anos.

Aos meus amigos da UFAM desde a graduação: Adriana, André e Ludimila, e aos amigos recém-adquiridos no mestrado: David e Juliana, pelas caronas, pausas para o café e almoço, pela amizade e pela ajuda nos momentos finais.

À Elienai, por todo o apoio nas questões administrativas. À UFAM e ao IComp, pela minha formação acadêmica e pela oportunidade concedida.

À FAPEAM, pela bolsa concedida, sem a qual este trabalho não poderia ser realizado.

Muito obrigado!

"A tarefa não é tanto ver aquilo que ninguém viu, mas pensar o que ninguém ainda pensou sobre aquilo que todo mundo vê."

Arthur Schopenhauer

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

A heterogeneidade das formas de modelar dados têm crescido nos bancos de dados e sistemas. Além disso, os esquemas de bancos de dados têm se tornado cada vez maiores e mais complexos. Assim, sempre que é necessária a integração de sistemas computacionais, é exigido um grande esforço para identificar estruturas em diferentes esquemas representando as mesmas entidades no mundo real [Bonifati and Velegrakis, 2011].

Dados dois (ou mais) esquemas pertencentes ao mesmo domínio, o problema de Casamento de Esquemas é definido como a tarefa de encontrar correspondências semânticas entre os elementos dos esquemas [Doan et al., 2000].

Por esquema, entendemos como qualquer conjunto de elementos conectados por alguma estrutura [Rahm and Bernstein, 2001]. Por exemplo, arquivos XML (*eXtensible Markup Language*), definições de esquema em XDR (*XML-Data Reduced*) e modelos em SQL (*Structured Query Language*).

Tradicionalmente, esta tarefa tem sido realizada manualmente por um profissional que geralmente tem conhecimentos detalhados sobre os esquemas que serão integrados e sobre o domínio em que eles estão inseridos. Entretanto, descobrir mapeamentos semânticos entre esquemas é uma tarefa tediosa, custosa e sujeita a erros.

Um exemplo da tarefa de casamento de esquemas (ou tarefa de *matching*) é mostrado na Figura 1.1, onde são ilustrados dois esquemas no domínio de ordem de compra e um possível casamento entre os seus elementos (setas pontilhadas).

Mapear todos os elementos nem sempre é possível. O exemplo mostra elementos que tem uma correspondência no outro esquema e note que nem todos os elementos podem ser casados. Além disso, há elementos que representam a mesma entidade mas são confundidos, pois, têm diferentes semânticas. Por exemplo: *Address* no primeiro esquema (derivado de *BillTo*) e *ShipAddress* no segundo esquema, ambos correspondem a um endereço, mas o primeiro é relacionado ao endereço de cobrança e o segundo é relacionado ao endereço de entrega da compra.

Para auxílio no processo, diversos sistemas têm sido lançados comercial-

Figura 1.1: Tarefa de casamento de esquemas. Setas pontilhadas representam as correspondências encontradas.

mente como o Biztalk [1] e Clio [Popa et al., 2002], e protótipos foram desenvolvidos através da pesquisa acadêmica [Madhavan et al., 2001, Beneventano et al., 2000, L. Palopol and Ursino, 1998, Jian et al., 2005]. Esses sistemas provêm correspondências entre os esquemas e o resultado gerado por um sistema é então avaliado por um especialista que corrige a resposta, seja adicionando correspondências não encontradas ou retirando correspondências incorretas.

Aplicações como integração de esquemas provêem uma representação unificada de diferentes esquemas, e apesar dos diversos estudos como [Conrad et al., 1997, Pottinger and Bernstein, 2003, Bilke and Naumann, 2005] esta tarefa se mantém difícil e as soluções ainda se apóiam na intervenção humana. Outro exemplo é o da construção de *data warehouses*, que consiste no projeto e implementação do modelo, ferramentas e facilidades para gerenciar e prover, em tempo hábil, informações precisas e claras para o apoio de tomada de decisão [Ballard et al., 1998]. Rahm et al. [2001] detalham outras aplicações como comércio eletrônico e processamento de consultas semânticas.

Todas essas aplicações envolvem em algum passo o problema do casamento de esquemas. Por exemplo, em *data warehouse* diversos dados de diferentes bancos de dados devem ser mapeados em um novo esquema. As diferenças entre as representações criaram a necessidade da realização de casamento de esquemas. Em transações de comércio eletrônico, os agentes participantes da transação trocam mensagens com formatação própria e estruturação diferente. Nesse contexto, realizar o mapeamento desses dados é uma tarefa crítica e exige rapidez.

O problema do casamento de esquemas é semelhante ao problema do casamento estável da área de Teoria da Computação: o problema consiste em parear elementos de dois conjuntos obedecendo a restrições como listas de preferências e estabilidade do pareamento [Kleinberg and Tardos, 2005]. A semelhança entre os problemas nos ajuda a compreender a dificuldade de se realizar o casamento entre esquemas de banco de dados.

Os métodos existentes utilizam diferentes tipos de informação para gerar os

[1] http://msdn.microsoft.com/en-us/library/aa547076.aspx

casamentos como propriedades referentes aos esquemas [Duchateau et al., 2009a, Do and Rahm, 2002, Beneventano et al., 2000], instâncias dos esquemas [Yang et al., 2008, Mesquita et al., 2007], além de combinar estratégias e utilizar a interação com o usuário através de interface gráfica [Aumueller et al., 2005]. Há ainda os métodos que utilizam técnicas de aprendizagem de máquina para realização da tarefa de casamento de esquemas como o *YAM–Yet Another Matcher* [Duchateau et al., 2009b], que utiliza uma abordagem supervisionada para a identificação de elementos correspondentes.

Em oposição a essas abordagens, propomos um método chamado *ALMa – **A**ctive **L**earning **Ma**tching*. O *ALMa* utiliza aprendizado ativo e consultas por comitê para a realização da tarefa do casamento de esquemas. Por aprendizado ativo, podemos entender como qualquer forma de aprendizado em que o método tem algum controle sobre as instâncias utilizadas para treinamento [Cohn et al., 1994]. Consulta por comitê é um algoritmo proposto por [Seung et al., 1992] onde um comitê de modelos de aprendizado é treinado com um mesmo conjunto de treino e este decide a próxima instância a ser incorporada no conjunto de treino pelo princípio do desacordo maximal.

Assim, definimos o objetivo do trabalho como: dado um conjunto de *matchers*, desenvolver um método que utilize a experiência do especialista aplicando a abordagem de aprendizado ativo para gerar soluções para o problema do casamento de esquemas.

Nos experimentos mostramos que uma abordagem heurística adotada pelo *baseline* (COMA [Do and Rahm, 2002]) não obtém bons resultados para um conjunto de testes desconhecido. Mostramos também que a abordagem proposta utilizando aprendizado ativo alcança bons níveis de revocação e valor de média-F igual a 0.64.

O restante desta dissertação está organizado como segue: No Capítulo 2 apresentamos o problema do casamento de esquemas. São apresentadas definições sobre o problema e são descritas algumas estratégias previamente propostas na literatura para a sua solução, além de pontuar a abordagem proposta que utiliza aprendizado ativo. No Capítulo 3 são apresentados os métodos baseados em aprendizado ativo, funcionamento, algoritmos e prós/contras. No Capítulo 4 são descritos os experimentos realizados, configurações utilizadas e resultados. No Capítulo 5 são feitas as considerações finais e comentários sobre trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo apresentamos o problema do casamento de esquemas de banco de dados, bem como desafios do problema e soluções propostas. Apresentamos também a correlação do problema de casamento de esquemas com o problema do casamento estável. Em seguida, exploramos a estratégia fixa com *matchers* utilizada por alguns métodos, discutimos a estratégia com aprendizado e por fim, apresentamos a estratégia que utiliza a estrutura de comitês e aprendizado ativo.

2.1 Casamento de Esquemas de Banco de Dados

O Casamento de Esquemas tem como objetivo identificar correspondências entre estruturas ou modelos, como esquemas de bancos de dados, mensagens formatadas em XML e ontologias. Resolver esse problema é a tarefa-chave para inúmeras aplicações como troca de dados, evolução de esquemas e virtualmente todos os tipos de integração de dados. Infelizmente, o alto grau de heterogeneidade semântica refletido nos diferentes esquemas faz do casamento de esquemas uma tarefa complexa [Bellahsene et al., 2011].

A Figura 2.1 mostra a representação simplificada de dois esquemas e os casamentos (ou correspondências) encontrados em linhas pontilhadas. O problema se mostra desafiador pois a tarefa de casamento pode conter elementos descritos de forma diferente (nesse caso por um acrônimo) e ainda assim serem correspondentes (*PurchaseOrder* e *PO*), elementos cuja semântica é parcialmente correspondente (*Name* representando o nome completo e *FName* somente o primeiro nome), elementos com nomes parecidos mas representando diferentes conceitos nos esquemas (*Address* representando o endereço de cobrança e *ShipAddress* representando o endereço de entrega) e elementos sem um correspondente no outro esquema (*Product*, *BillTo* e *Customer*).

O casamento de esquemas é uma das operações básicas do processo de integração de dados [Bernstein and Melnik, 2004]. Geralmente é seguida pelo mapeamento de dados, tarefa de encontrar funções que mapeiam dados de esquemas heterogêneos, passando dados formatados em um esquema para outro formato. Esta tarefa resulta em diversas

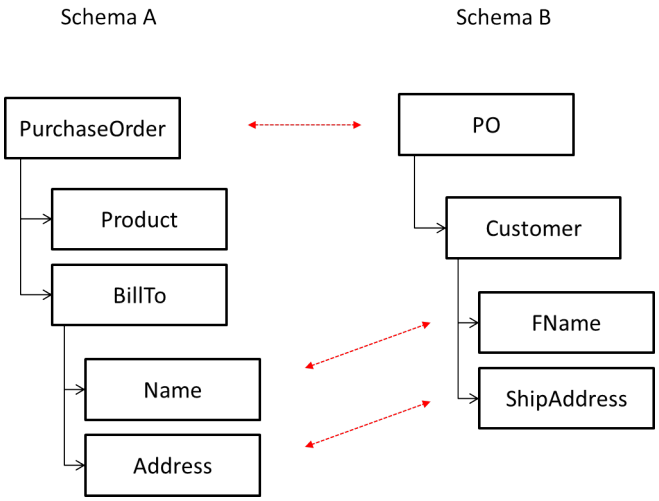


Figura 2.1: Dois esquemas e os casamentos (setas pontilhadas) possíveis.

aplicações, tais como geração de esquemas globais, reescrita de consultas e eliminação de dados duplicados [Gal, 2006].

Apesar dos diversos protótipos e métodos lançados ao longo dos anos, eles não conseguem sozinhos resolver esta etapa em um trabalho de integração. Sendo assim, a validação de um especialista ainda se faz necessária seja para adicionar correspondências que não foram encontradas, seja para retirar correspondências incorretas.

Os métodos utilizam diferentes tipos de informação para descobrir as correspondências. Informações como similaridade dos nomes, descrições, tipos de dados, estrutura, padrões e intervalos de valores são alguns dos dados utilizados mas, nem sempre esses dados estão disponíveis. Diferentes formas de utilizar essas informações e como são combinadas são relatadas em [Rahm and Bernstein, 2001].

2.2 O Problema do Casamento Estável

Semelhante ao problema do casamento de esquemas de banco de dados temos o problema do casamento estável da área de teoria da computação. O problema do casamento estável originou-se, em parte, com os matemáticos David Gale e Lloyd Shapley [Kleinberg and Tardos, 2005] e a versão clássica do problema pode ser definida como: Dado um conjunto de homens e mulheres e associada a cada elemento uma lista de preferências de parceiros. Deve-se casar os elementos de forma que a situação a seguir seja evitada:

- um homem h_y preferir uma outra mulher m_x à sua atual esposa; e
- a mulher m_x preferir o homem h_y a seu atual marido.

Se essa condição é satisfeita para todos os casais formados, dizemos que o casamento é estável. Uma década antes deste problema ser proposto, uma outra versão

(desconhecida por Gale e Shapley) tinha sido estudada, o problema similar tratava de atribuir hospitais a residentes [Kleinberg and Tardos, 2005].

Gale e Shapley desenvolveram um algoritmo para o problema clássico de casamento. Esse algoritmo gera a solução ótima e tem complexidade $O(n^2)$, estando, portanto, o problema em P.

O problema do casamento de esquemas pode ser mapeado para o problema do casamento estável da seguinte forma:

- Elementos dos esquemas formam os conjuntos de homens e mulheres;
- As similaridades entre elementos denotam as preferências;
- Cada elemento do esquema, a princípio, é casado com um elemento do outro esquema.

E as particularidades do problema do casamento de esquemas podem ser relacionadas a versões mais complexas do casamento estável como:

- Valores de similaridade iguais para dois ou mais pares de elementos podem ser interpretados como empate na lista de preferências;
- Mais de uma medida similaridade pode ser usada, ou seja, múltiplas listas de preferências;
- Valores de similaridades nulos produziriam listas de preferências incompletas;
- Elemento de um esquema tendo mais de um correspondente no outro esquema, denotando uma poligamia;
- Restrições como tipo de dados impedem que algumas correspondências sejam possíveis, assemelhando-se a listas de pares proibidos.

Qualquer uma dessas particularidades mencionadas configura uma versão do casamento estável sem solução computacional eficiente, colocando-o na classe de problemas NP-difíceis [Manlove et al., 2002].

2.3 *Matchers*

Os *matchers* são funções que recebem como entrada um par de elementos de diferentes esquemas e estimam um valor de similaridade entre os elementos, segundo algum critério. Um *matcher* é o operador básico de diversos métodos propostos para o problema do casamento de esquemas [Madhavan et al., 2001, Do and Rahm, 2002, Aumueller et al., 2005, Chukmol et al., 2005, Mesquita et al., 2007, Cruz et al., 2009,

Duchateau et al., 2009a], o valor dado por um matcher deve refletir a similaridade entre o par de elementos analisado segundo o critério referido, geralmente esse valor é normalizado entre $[0, 1]$ sendo 0 o valor que denota nenhuma similaridade e 1 denotando total similaridade. Um exemplo de resultado de um *matcher* pode ser visto na Tabela 2.1 que mostra a matriz de similaridade gerada pelo matcher *NamePath*.

NamePath	PO	Customer	Fname	ShipAddress
PurchaseOrder	0.80	0.30	0.28	0.16
Product	0.50	0.20	0.12	0.22
BillTo	0.60	0.33	0.12	0.20
Name	0.30	0.65	1.00	0.25
Address	0.30	0.22	0.20	1.00

Tabela 2.1: Similaridade por *NamePath*.

Uma vasta quantidade de *matchers* tem sido utilizada pelos métodos e diferentes tipos de *matchers* são utilizados. Existem *matchers* linguísticos que utilizam dados como nomes e descrições, *matchers* baseados em restrições que utilizam tipos de dados e a topologia do grafo do esquema, *matchers* que utilizam as instâncias do esquema e *matchers* híbridos que são compostos por mais de um *matcher*.

2.4 Estratégias

Os *matchers* representam as funções básicas dos métodos de casamento de esquemas e estes geram as matrizes de similaridade. Os métodos publicados utilizam diferentes estratégias para manipular essas matrizes e escolher as correspondências corretas. Apresentamos a seguir duas estratégias: a estratégia fixa, onde os métodos utilizam uma heurística para selecionar as correspondências válidas e, no contexto de aprendizagem de máquina, seria considerado um método não-supervisionado; e a estratégia com aprendizado que utiliza algoritmos de aprendizagem de máquina e dados de treino. Em adição, mostramos uma alternativa que utiliza algoritmos de aprendizagem de máquina com aprendizado ativo.

2.4.1 Estratégia Heurística

Existem diversas formas de estabelecer as correspondências entre esquemas, alguns métodos utilizam heurísticas que manipulam os dados das matrizes de similaridade e assim determinam que correspondências são retornadas como resposta.

Soluções propostas em [Shiang et al., 2008, Melnik et al., 2002] utilizam semelhanças entre os grafos das estruturas dos esquemas de forma que as similaridades entre os elementos são propagadas para os seus predecessores/sucessores. Os protótipos COMA [Do and Rahm, 2002] (e seu sucessor COMA++ [Aumueller et al., 2005]) e

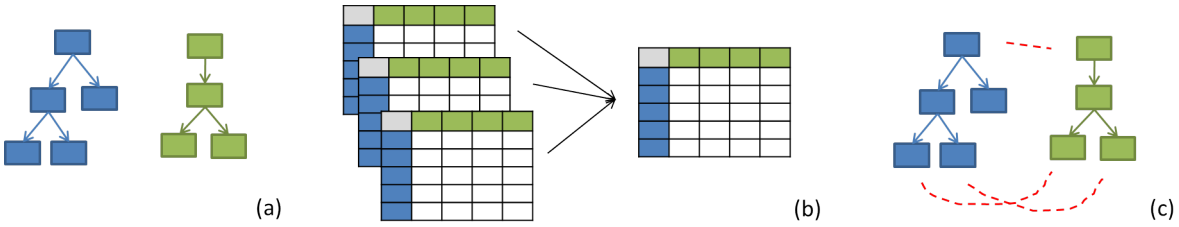


Figura 2.2: Funcionamento do COMA. (a) Representação dos esquemas. (b) Agregação das matrizes de similaridade. (c) Resolução das correspondências.

AgreementMaker [Cruz et al., 2009] utilizam uma heurística de seleção a partir das matrizes de similaridades.

Para ilustrar melhor esse tipo de estratégia apresentamos o funcionamento do COMA (que será utilizado como *baseline* nesse trabalho). Dados dois esquemas (Figura 2.2(a)), um conjunto de *matchers* é aplicado sobre os elementos dos esquemas e uma matriz de similaridade é montada para cada *matcher*, os valores das matrizes são então agregados em uma única matriz (Figura 2.2(b)), um critério de seleção é utilizado sobre os valores dessa matriz e então as correspondências são geradas (Figura 2.2(c)).

A biblioteca de *matchers* utilizada no COMA contém sete *matchers* simples e cinco *matchers* híbridos. *Matchers* simples utilizam apenas uma função sobre os elementos dos esquemas enquanto que os *matchers* híbridos utilizam combinações de mais de um *matcher* simples, além de técnicas como a divisão dos elementos em *tokens*. O critério de avaliação de cada *matcher* utilizado está descrito na Tabela 2.2, detalhes sobre o modo como os *matchers* são combinados podem ser consultados em [Do and Rahm, 2002].

	Matcher	Estratégia
Simples	Affix	Procura por prefixos/sufixos em comum
	2-gram	Comparação por bigramas
	3-gram	Comparação por trigramas
	Soundex	Calcula a similaridade fonética
	EditDistance	Calcula a similaridade de acordo com a distância *Levenshtein*
	Synonym	Utiliza um dicionário de sinônimos
	DataType	Avalia a compatibilidade entre os tipos de dados
Híbridos	Name	Combina os *matchers* Affix, 3-gram e Synonym
	NamePath	Utiliza todo o caminho do elemento desde a raiz
	TypeName	Combina os *matchers* DataType e Name
	Children	Combinando as similaridades dos filhos dos elementos
	Leaves	Utiliza somente os elementos-folha

Tabela 2.2: Biblioteca de *matchers* disponíveis no COMA.

Apesar dos vários *matchers* disponíveis, o COMA utiliza apenas os cinco *matchers* híbridos citados, ou seja, apenas cinco matrizes de similaridade são criadas e os valores destas são agregados em uma matriz através da média. Para ilustrar essa agregação,

NamePath	PO	Customer	Fname	ShipAddress
PurchaseOrder	0,80	0,30	0,28	0,16
Product	0,50	0,20	0,12	0,22
BillTo	0,60	0,33	0,12	0,20
Name	0,30	0,65	1,00	0,25
Address	0,30	0,22	0,20	1,00

Levenshtein	PO	Customer	Fname	ShipAddress
PurchaseOrder	0,57	0,62	0,00	0,58
Product	0,78	0,49	0,00	0,48
BillTo	0,00	0,52	0,00	0,41
Name	0,00	0,00	0,93	0,00
Address	0,00	0,00	0,56	0,00

Média	PO	Customer	Fname	ShipAddress
PurchaseOrder	0,69	0,46	0,14	0,37
Product	0,64	0,35	0,06	0,35
BillTo	0,30	0,43	0,06	0,31
Name	0,15	0,33	0,97	0,13
Address	0,15	0,11	0,38	0,50

Figura 2.3: Agregação de matrizes de similaridade através da média.

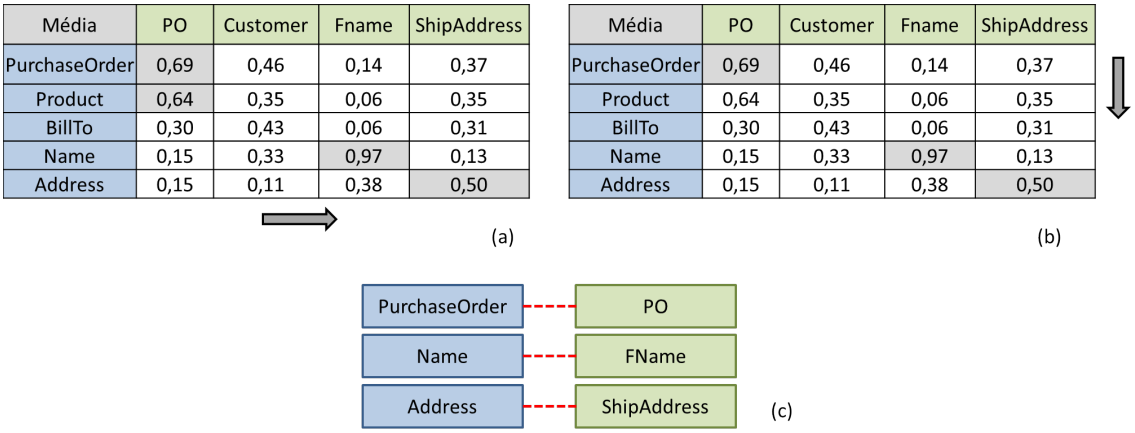


Figura 2.4: Heurística de seleção das correspondências feita pelo COMA. (a) Análise por linha. (b) Análise por coluna. (c) Correspondências encontradas.

considere as matrizes da Figura 2.3, para cada célula, a média é retirada e uma matriz agregada é formada.

Depois de obter uma matriz agregada, um processo de seleção é aplicado. Somente os valores iguais ou acima de 0.50 são considerados. Os autores afirmam que valores abaixo desse limite inferior não representam correspondências válidas. Primeiro, a matriz é analisada linha por linha (Figura 2.4(a)), são selecionados na matriz o elemento que tem o maior valor e aqueles que distam do maior elemento por, no máximo, 0.02 (a intenção dos autores é melhorar a revocação selecionando correspondências com valor de similaridade próximo ao valor maximal). De forma semelhante, a matriz é analisada coluna por coluna (Figura 2.4(b)). Se um elemento da matriz foi selecionado nas duas etapas (e tem valor maior ou igual a 0.50), então ele é selecionado como uma correspondência válida e é retornado como resposta (Figura 2.4(c)).

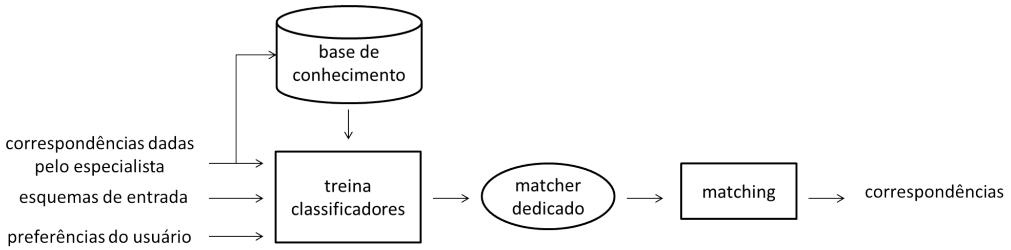


Figura 2.5: Arquitetura do YAM [Duchateau et al., 2009a].

2.4.2 Estratégia com Aprendizado Supervisionado

Diversas soluções utilizam algoritmos de aprendizado para estabelecer correspondências. Essa abordagem tem a desvantagem de utilizar dados para treino dos classificadores e nem sempre estes dados estão disponíveis. Estes dados geralmente são provenientes de execuções anteriores do método que ficam armazenadas e da experiência do usuário que rotula as correspondências.

Entre os métodos que utilizam a abordagem supervisionada, podemos citar o Clio [Popa et al., 2002] que utiliza um classificador de atributos para aprender as possíveis correspondências. O trabalho de Li et al. [2005] utiliza um algoritmo de agrupamento para geração de grupos (*clusters*) e redes neurais sobre os *clusters* para aprender as correspondências corretas.

Demonstramos a estratégia supervisionada com o YAM [Duchateau et al., 2009a]. A arquitetura do YAM (*Yet Another Matcher*) é mostrada na Figura 2.5. O YAM gera um modelo para escolha de correspondências diferente para cada tarefa de casamento. Isto porque o YAM utiliza diversos algoritmos de aprendizado para, dada uma tarefa de casamento, treinar diversos classificadores e utilizar o melhor classificador da fase de treino para gerar as correspondências.

O YAM recebe como dados de entrada correspondências que são dadas pelo usuário, os esquemas de entrada e o usuário também pode explicitar preferência sobre precisão e revocação. A base de conhecimento contém classificadores, funções de similaridade e correspondências provenientes de execuções anteriores. Os classificadores são treinados com os dados da base de conhecimento e o classificador que alcança os melhores resultados é escolhido para realizar a tarefa de casamento. Uma vez escolhido o classificador, este é usado junto com os esquemas de entrada e gera as correspondências.

2.4.3 Estratégia com Aprendizado Ativo

Estratégias fixas utilizando heurísticas podem não ser eficientes para diferentes domínios de aplicação, estratégias supervisionadas dependem da obtenção de dados rotulados cuja obtenção é difícil. Diversos métodos utilizam o conhecimento do especialista para melhorar os resultados gerados, seja este adicionando correspondências

não encontradas ou removendo correspondências incorretas [Bernstein et al., 2006, Wagner et al., 2011].

Como alternativa, apresentamos uma estratégia que utiliza aprendizado ativo para realizar a tarefa de casamento. Na abordagem de aprendizado ativo, o método de aprendizado solicita ao especialista quais instâncias devem ser rotuladas, ou seja, o especialista é solicitado para dizer se uma correspondência é verdadeira ou falsa. Uma instância é escolhida para rotulação através do algoritmo de consultas por comitê: um comitê de modelos de aprendizado vota sobre o rótulo de uma instância, se uma instância divide as opiniões dos membros do comitê, essa instância é apresentada ao especialista para rotulação. Note que essa abordagem difere das citadas anteriormente por invocar o especialista para que este rotule instâncias, em oposição ao especialista procurar erros no conjunto de respostas do método.

Utilizando o aprendizado ativo, pretendemos reduzir a intervenção do usuário comparada a uma abordagem supervisionada e utilizamos um algoritmo de aprendizado sobre a biblioteca de *matchers* para aprender classificadores para cada tarefa de casamento, no intuito de evitar que o método seja eficaz somente para um domínio específico. O método é apresentado no Capítulo 3.

Capítulo 3

Métodos Propostos para o Casamento de Esquemas

Neste capítulo apresentamos o *ALMa* – ***A****ctive* ***L****earning* ***Ma****tching*. O método desenvolvido utiliza a abordagem de aprendizado ativo para gerar correspondências entre esquemas. O uso do aprendizado ativo possibilita a diminuição da quantidade de treino necessária em comparação com a abordagem supervisionada e desvincula o método de uma estratégia fixa que pode não ser eficaz para todos os domínios.

Diferente do aprendizado supervisionado onde um conjunto de instâncias é rotulado previamente, no aprendizado ativo, o método escolhe quais instâncias devem ser rotuladas e quando devem ser rotuladas [Beygelzimer et al., 2009]. O *ALMa* utiliza uma estrutura de comitê para requisitar essas instâncias do especialista. Um comitê é formado por diferentes modelos de um algoritmo de aprendizado de modo que cada modelo tenha uma opinião independente sobre uma instância. O ganho de informação de uma consulta (relacionado ao rótulo de uma instância) é alto quando o desacordo entre os membros do comitê é maximizado [Seung et al., 1992].

O modelo de aprendizado utilizado foi árvore de decisão construída de acordo com o algoritmo C4.5 [Quinlan, 1993]. As árvores de decisão foram escolhidas por serem amplamente utilizadas na abordagem com comitês, por serem simples e o modelo é flexível aos diferentes conjuntos de entrada [Bishop, 2006].

Considere as matrizes de similaridade por *NamePath* e *Levenshtein* mostradas na Figura 2.3. Considere também a árvore de decisão da Figura 3.1. A decisão de uma árvore é obtida percorrendo-se a árvore a partir de sua raiz, logo, a classe de uma instância é dada pela árvore nos nós-folha. No exemplo, a árvore atribuiu a classe `VERDADEIRO` para quatro pares de elementos.

A idéia básica da abordagem é treinar um conjunto de árvores de decisão, formar um comitê com um subconjunto da população de árvores e deixar esse conjunto decidir sobre o rótulo de uma instância. Para o método *ALMa*, uma instância é descrita por um par de elementos (um elemento de cada esquema da tarefa de casamento) e os

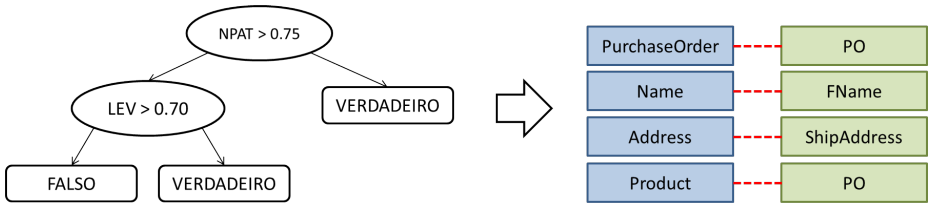


Figura 3.1: Exemplo de árvore de decisão e as correspondências que ela julga verdadeira segundo os *matchers* da Figura 2.3.

valores de similaridade dados pelos *matchers*, adicionalmente, cada instância tem uma classe (ou rótulo) atribuída. Se a correspondência entre os elementos não é válida, a instância tem o rótulo `FALSO` associado, caso contrário, ela recebe o rótulo `VERDADEIRO`. Um exemplo é mostrado na Tabela 3.1.

Instância		Matchers				
Esquema-1	Esquema-2	3GRAM	NPATH	LEAVE	CHILD	CLASS
PurchaseOrder	Po	0.077	0.083	0.540	0.147	`VERDADEIRO`
PurchaseOrder	Customer	0.192	0.073	0.543	0.223	`FALSO`
PurchaseOrder	FName	0.103	0.056	0.307	0.150	`FALSO`
PurchaseOrder	ShipAddress	0.077	0.096	0.307	0.150	`FALSO`
			...			
Address	ShipAddress	0.545	0.366	0.767	0.767	`VERDADEIRO`

Tabela 3.1: Exemplo de instâncias e valores dados pelos *matchers*

Seja o exemplo da Tabela 3.2. Um comitê formado por seis árvores é utilizado para rotular quatro instâncias, cada árvore opina sobre cada instância e a maioria dos votos decide a opinião do comitê. A decisão do comitê chega a um estado de `EMPATE` quando as opiniões dos membros são divididas. Um valor de confiança pode ser associado à decisão do comitê.

Neste trabalho propomos duas versões do método *ALMa*: o *ALMa-OC* (*Open Committee*) e o *ALMa-CC* (*Closed Committee*). As versões do *ALMa* apresentadas diferem basicamente no momento em que os rótulos finais das instâncias são decididos pelo comitê, ou seja, quando o comitê decide definitivamente que uma correspondência é válida.

O funcionamento básico para as duas versões inicia-se com a escolha de algumas instâncias para treino (Figura 3.2(a)), essas instâncias são repassadas ao espe-

	T1	T2	T3	T4	T5	T6	Decisão
I1	F	V	V	F	F	F	F
I2	V	V	V	F	V	F	V
I3	V	F	V	V	F	F	E
I4	F	V	F	F	F	F	F

Tabela 3.2: Decisão do comitê (não-ponderada).

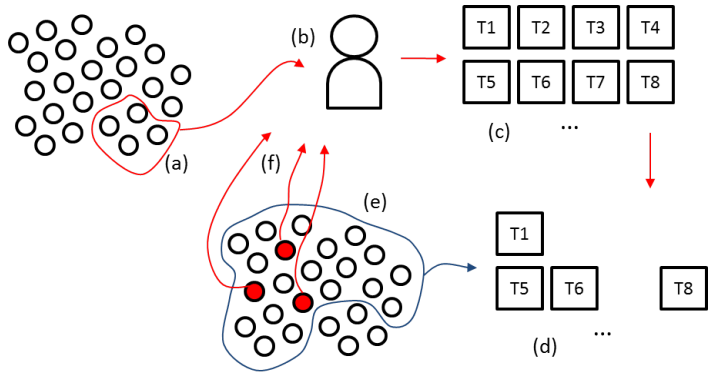


Figura 3.2: Funcionamento básico do *ALMa*. (a) Conjunto de instâncias de treino. (b) Especialista consultado. (c) Conjunto de árvores de decisão. (d) Comitê formado. (e) Conjuto de instâncias de teste. (f) Novas instâncias a serem rotuladas.

cialista/oráculo para receberem o rótulo correto (Figura 3.2(b)). As árvores de decisão são treinadas (Figura 3.2(c)) e um subconjunto da população de árvores forma o comitê da rodada (Figura 3.2(d)). O comitê avalia as instâncias não-rotuladas (Figura 3.2(e)) e as instâncias que dividiram a opinião do comitê são selecionadas para serem rotuladas na próxima iteração (Figura 3.2(f)).

Nas seções seguintes, apresentamos as intuições e os algoritmos de duas versões do *ALMa* e em seguida fazemos uma comparação entre as abordagens.

3.1 *ALMa-OC*

O *ALMa-OC* (***A**ctive **L**earning **Ma**tching – **O**pen **C**ommittee*) é baseado na idéia de criar diferentes comitês para avaliar as instâncias e encontrar correspondências. Dizemos que o comitê utilizado é aberto, no sentido de que a cada iteração um comitê é escolhido e este pode definir o rótulo final de uma instância. O objetivo é encontrar comitês que reconheçam diferentes padrões de instâncias verdadeiras (de acordo com os valores dados pelos *matchers*).

O processo é apresentado no Algoritmo 1. O método inicia com a escolha de *kClusters* instâncias para que estas sejam rotuladas pelo especialista (detalhes no Algoritmo 2) . Essas instâncias devem, preferencialmente, ser diferentes entre si (em relação aos valores dados pelos *matchers*), dessa forma diminuimos a chance de um aprendizado viciado (*overfitting*) por parte da população de árvores de decisão. Para evitar este cenário, as instâncias são agrupadas utilizando o algoritmo *KMeans* [Witten et al., 2011] e os grupos são formados por instâncias que têm valores semelhantes dados pelos matchers. Após o agrupamento, uma instância aleatória é retirada de cada um dos grupos (*clusters*) formados. Esse conjunto forma o primeiro conjunto de treino (Figura 3.3(a)).

No próximo passo, o especialista é convocado para rotular as instâncias de treino

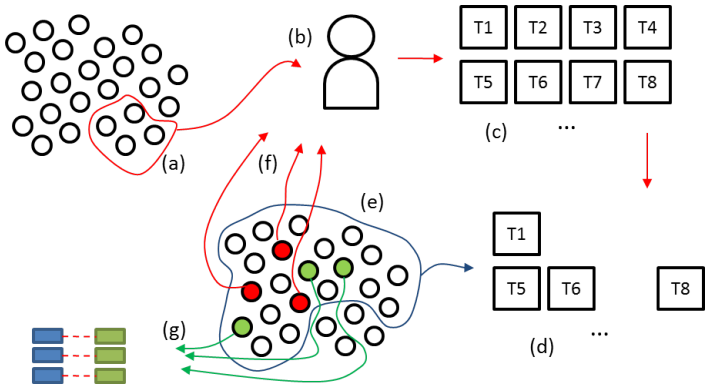


Figura 3.3: Funcionamento do *ALMa-OC*: (a) Conjunto de instâncias de treino selecionadas. (b) Especialista invocado para rotulação das instâncias de treino. (c) População de árvores de decisão. (d) Comitê formado (subconjunto da população de árvores de decisão). (e) Conjuto de instâncias de teste. (f) Instâncias que geraram divergência de opiniões entre os membros do comitê são rotuladas na próxima iteração. (g) Instâncias classificadas positivas (com alto grau de certeza) são devolvidas como resposta e retiradas do processo.

(LS) selecionadas aleatoriamente dos grupos gerados pelo *KMeans* (Figura 3.3(b)).

No intuito de criar diferentes árvores na população, cada árvore pode utilizar apenas um subconjunto dos *matchers* disponíveis (por exemplo, a árvore da Figura 3.1 utiliza dois *matchers: TypeName* e *Levenshtein*). Na primeira iteração, existem árvores que utilizam somente os valores de um *matcher*.

No *loop* principal, as árvores são treinadas de acordo com o algoritmo C4.5 [Quinlan, 1993]. Cada árvore recebe um peso WT que reflete a acurácia da árvore sobre a avaliação nas instâncias de treino. A acurácia de uma árvore é calculada dividindo-se a quantidade de instâncias corretas pela quantidade de instâncias avaliadas pela árvore (Figura 3.3(c)).

As árvores com maior peso WT são nomeadas membros de um comitê. O comitê deve ter o tamanho mínimo $minTamC$ mas pode incluir árvores que tenham valor de WT igual ao da árvore com menor peso presente no comitê (Figura 3.3(d)).

As árvores de decisão emitem opiniões sobre as instâncias (os membros do comitê decidem o rótulo da instância através de voto ponderado por WT), e um valor de confiança WC e um valor de desacordo WD são associados a essa votação (detalhes no Algoritmo 3) (Figura 3.3(e)). O peso WC reflete a confiança que o comitê tem sobre o rótulo de uma instância e o valor WD indica o desacordo provocado entre os membros do comitê.

Os pesos WT das árvores são atualizados de acordo com o Algoritmo 4 [de Freitas et al., 2010]. As árvores que classificam instâncias como `VERDADEIRO` junto com o comitê têm o peso aumentado, as árvores são penalizadas quando classificam como `FALSO` instâncias que o comitê julgou como `VERDADEIRO`.

Finalizando a iteração, novas instâncias são escolhidas para compor o conjunto de

instâncias de treino (Figura 3.3(f)). As instâncias escolhidas são as que tiveram maior desacordo na votação do comitê (no máximo $instPorIt$ instâncias por iteração). O desacordo provocado no comitê por uma instância (o peso WD), é calculado pela soma dos pesos das árvores que votaram na classe minoritária dividido pela soma dos pesos de todas as árvores. O peso WC é a soma dos pesos das árvores que votaram na classe majoritária menos a soma dos pesos das árvores que votaram na classe minoritária dividido pela soma dos pesos que votaram na classe majoritária. O peso WC é maior quando as árvores que votam na classe minoritária são poucas ou têm peso baixo. O valor desse peso é normalizado para o intervalo $[0.0 - 0.8]$, pois, apenas instâncias rotuladas pelo especialista possuem o peso WC máximo.

As instâncias que, pelos votos do comitê, foram classificadas como `VERDADEIRO` e possuem uma confiança associada maior que $minConf$ são mostradas como resposta e retiradas do conjunto AS (Figura 3.3(g)).

A Tabela 3.3 mostra as variáveis utilizadas pelos algoritmos apresentados. Detalhes sobre a parametrização do método são exploradas no Capítulo 4.

Algoritmo 1: ALMa-OC

Entrada: AS,$kClusters$,$minTamC$,$minConf$,$instPorIt$
Saída: R

1 $LS \Leftarrow escolheInstanciasKMeans(kClusters)$;
2 $DT_t \Leftarrow escolheMatchers(matchers)$;
3 $rotulo(LS)$; `/* rotulação de instâncias */`
4 $k \Leftarrow 0$;
5 **enquanto** $k \leq maxIt$ **faça**
6 $TS \Leftarrow AS - LS$;
7 $C4.5(DT, LS)$; `/* treina as árvores de decisão */`
8 $WT_t \Leftarrow acuracia(DT_t, LS)$;
9 $C_k \Leftarrow decideComite(DT, WT, minTamC)$;
10 $avaliaInstâncias(DT, AS)$; ; `/* árvores emitem opinião sobre as instâncias */`
11 $WD, WC \Leftarrow atualizaConfiancaDesacordo(CVote)$;
12 $atualizaPesoArv(DT, AllVote, C_k)$;
13 $R \Leftarrow R + eliminaInst(minConf, CVote)$;
14 $LS \Leftarrow LS + selecionaInst(WD, instPorIt)$;
15 $k++$;
16 **fim enquanto**
17

3.2 *ALMa-CC*

A abordagem *ALMa-CC* (***A**ctive **L**earning **Ma**tching – **C**losed **C**ommittee*) objetiva treinar um único comitê para definir o rótulo final de uma instância. Ao decorrer

Tabela de variáveis

DT	Conjunto de árvores de decisão
AS	*All Set* - conjunto de instâncias
LS	*Learning Set* - conjunto de instâncias de treino
TS	*Testing Set* - conjunto de instâncias de teste
WT	Peso de uma árvore
WD	Desacordo do comitê sobre uma instância
WC	Confiança do comitê sobre o rótulo de uma instância
C	Comitê da iteração
$CVote$	Decisão do comitê sobre o rótulo de uma instância
$AllVote$	Voto de uma árvore sobre o rótulo de uma instância
Cl	*Clusters* gerados pelo KMeans
R	Conjuto de respostas do método
AVG	Média dos valores dados pelos *matchers*
TOP	Conjunto *Top* - contém as instâncias com maiores valores AVG
BOT	Conjunto *Bottom* - contém as instâncias com menores valores AVG
KM	Conjunto *KMeans* - contém as instâncias selecionadas após o agrupamento por *KMeans*
k	Iteração corrente
$matchers$	Conjunto de *matchers* disponíveis
$kClusters$	Quantidade de *clusters* gerados pelo *KMeans*
$minTamC$	Menor tamanho de comitê permitido
$minConf$	Menor valor de confiança para que um rótulo final seja atribuído
$instPorIt$	Quantidade máxima de instâncias rotuladas pelo especialista por iteração
$maxIt$	Quantidade máxima de iterações
$qtTB$	Quantidade de instâncias nos conjuntos *Top/Bottom*
$maxRotPorIt$	Quantidade máxima de instâncias rotuladas pelo especialista por iteração
$maxDisag$	Desacordo máximo permitido
Índice sobrescrito t	Denota uma árvore t
Índice sobrescrito i	Denota uma instância i
Índice sobrescrito k	Denota uma iteração k

Tabela 3.3: Variáveis utilizadas pelos algoritmos.

Algoritmo 2: escolheInstanciasKMeans

Entrada: $kClusters$
Saída: LS

1 $Cl \Leftarrow KMeans(kClusters, AS)$;
2 **para cada** Cl_j **faça**
3 $\quad LS \Leftarrow LS + instanciaAleatoria(Cl_j)$; ; `/* escolhe uma instância aleatória desse cluster */`
4 **fim para cada**

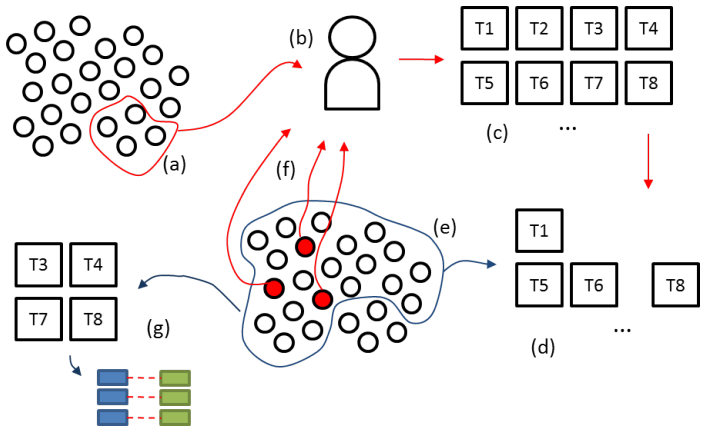


Figura 3.4: Funcionamento do *ALMa-CC*: (a) Conjunto de instâncias de treino selecionadas. (b) Especialista invocado para rotulação das instâncias de treino. (c) População de árvores de decisão. (d) Comitê formado (subconjunto da população de árvores de decisão). (e) Conjuto de instâncias de teste. (f) Instâncias que geraram divergência de opiniões entre os membros do comitê são rotuladas na próxima iteração. (g) Quando o comitê não tem mais dúvidas sobre nenhuma instância, atribui o rótulo final e, as instâncias classificadas positivas são devolvidas como resposta.

do aprendizado, o novo comitê gerado tenta descobrir os padrões de diferença entre as instâncias verdadeiras e falsas baseadas nas instâncias de treino, ou seja, o comitê aprende quais *matchers* são úteis para distinguir as instâncias corretas das incorretas. Comitês iniciais tendem a descordar mais devido a pouca quantidade de instâncias de treino, com a evolução das iterações, as instâncias que dividem o comitê são repassadas ao especialista para que recebam um rótulo e as árvores são retreinadas. O método pára quando o comitê converge e instâncias duvidosas não são mais encontradas.

O Algoritmo 5 mostra o funcionamento do método. Semelhante ao *ALMa-OC*, o método utiliza instâncias aleatórias de conjuntos agrupados por *KMeans*, além dessas instâncias, o método seleciona instâncias para o conjunto *Top* e *Bottom* (Figura 3.4(a)). A intuição é selecionar instâncias que certamente são verdadeiras (*Top*) e instâncias certamente falsas (*Bottom*) [de Freitas et al., 2010] de modo a evitar um viés para alguma das classes pelas árvores geradas.

Cada instância recebe um valor AVG que é a média dos valores dados pelos matchers, instâncias com média próxima de 1 provavelmente são da classe `VERDADEIRO`,

Algoritmo 3: atualizaConfiancaDesacordo

Entrada: $CVote$
Saída: WD, WC
/* Para este algoritmo somente árvores pertencentes ao comitê são utilizadas */
$WT_{(voto=Verd)}$; /* peso de uma árvore que votou verdadeiro */
$WT_{(voto=Falso)}$; /* peso de uma árvore que votou falso */
para cada AS_i **faça**
  se $CVote_i =$ *VERDADEIRO* **então**
    $WD_i = \frac{\sum WT_{(voto=Falso)}}{\sum WT}$;
    $WC_i = \frac{\sum WT_{(voto=Verd)} - \sum WT_{(voto=Falso)}}{\sum WT_{(voto=Verd)}}$;
  senão
    $WD_i = \frac{\sum WT_{(voto=Verd)}}{\sum WT}$;
    $WC_i = \frac{\sum WT_{(voto=Falso)} - \sum WT_{(voto=Verd)}}{\sum WT_{(voto=Falso)}}$;
  fim se
  ; /* se uma instância foi rotulada pelo especialista ela recebe peso de confiança máximo e desacordo nulo */
  se $AS_i \in LS$ **então**
    $WD_i = 0$;
    $WC_i = 1$;
  fim se
fim para cada

enquanto que as instâncias com média próxima de 0 provavelmente são da classe FALSO.

O conjunto *Top*, *Bottom* e as instâncias aleatórias retiradas dos grupos gerados pelo *KMeans* formam o conjunto de treino inicial do método. Esse conjunto é submetido ao especialista para rotulação (Figura 3.4(b)).

Também de forma semelhante à primeira versão, as árvores recebem um subconjunto aleatório dos *matchers* e são treinadas utilizando o algoritmo $C4.5$[Quinlan, 1993]. A fase inicial termina com o peso WT sendo atribuído às árvores de acordo com a acurácia obtida no conjunto de treino (Figura 3.4(c)).

No passo seguinte é feita a escolha do comitê. As árvores com maior valor WT são eleitas membros do comitê. O comitê deve ter um número mínimo de membros porém, as árvores com valores WT iguais ao da árvore membro do comitê com menor valor WT também são incluídas no comitê da rodada (Figura 3.4(d)).

O comitê escolhido emite opiniões sobre as instâncias (Figura 3.4(e)) e aquelas que dividem as opiniões dos membros são rotuladas pelo especialista no próximo passo (respeitando o limite de instâncias estabelecido por $maxRotPorIt$). Uma instância recebe o estado de dúvida ou empate se o desacordo WD associado a votação sobre o rótulo da instância ultrapassa o valor definido por $maxDisag$ (Figura 3.4(f)).

Os pesos das instâncias são atualizados segundo o Algo-

Algoritmo 4: atualizaPesosArv

Entrada: $DT, AllVote, C_k$
Saída: WT

```
1  para cada DT_t faça
2      WT_t ⇐ 0;
3      se DT_t ∈ C_k então
           ;                                          /* árvores do comitê */
4          se AllVote_{t,i} = VERDADEIRO então
5              WT_t ⇐ WT_t + WC_i
6          fim se
7          se AllVote_{t,i} = FALSO e CVote_i = VERDADEIRO então
8              WT_t ⇐ WT_t − WC_i
9          fim se
10     senão
11         se CVote = VERDADEIRO então
12             se AllVote_{t,i} = VERDADEIRO então
13                 WT_t ⇐ WT_t + WC_i
14             senão
15                 WT_t ⇐ WT_t − WC_i
16             fim se
17         fim se
18     fim se
19 fim para cada
```

ritmo 6 [de Freitas et al., 2010], quanto maior a certeza do comitê sobre o rótulo de uma instância, mais próximo de -1 (`FALSO`) ou 1(`VERDADEIRO`) o peso estará. As árvores que não fazem parte do comitê são retreinadas e estas avaliam as instâncias de teste. Os pesos das árvores são atualizados utilizando os rótulos dados pelo comitê (detalhes no Algoritmo 4).

Se nenhuma instância gerou dúvida ao comitê na iteração corrente, significa que o comitê convergiu e o algoritmo passa para uma última avaliação das instâncias de teste pelo comitê. As instâncias classificadas pelo último comitê como sendo da classe `VERDADEIRO` são retornadas no conjunto R (Figura 3.4(g)).

Detalhes sobre a parametrização do método são exploradas no Capítulo 4.

3.3 Diferenças entre as abordagens

Apesar de compartilhar boa parte do algoritmo, as duas abordagens apresentam características únicas que as diferem. Sendo assim, sumarizamos as diferenças entre elas na Tabela 3.4.

Em oposição à versão *ALMa-CC* que utiliza apenas um comitê para classificar as instâncias, a versão *ALMa-OC* alcança uma maior variedade de instâncias (em relação

Algoritmo 5: ALMa-CC

Entrada: $AS, kClusters, qtTB, minTamC, maxRotPorIt, maxDisag$
Saída: R

```
1  KM ⇐ escolheInstanciasKMeans(kClusters);
2  TOP ⇐ escolheInstanciasTop(AVG, qtTB);
3  BOT ⇐ escolheInstanciasBottom(AVG, qtTB);
4  LS ⇐ TOP + BOT + KM;
5  DT_t ⇐ escolheMatchers(matchers);
6  rotulo(LS); ;                          /* rotulação de instâncias */
7  C4.5(DT_t, LS);                    /* treina as árvores de decisão */
8  WT_t ⇐ acuracia(DT_t, LS);
9  k ⇐ 0;
10 repita
11 |  C_k ⇐ decideComite(DT, WT, minTamC);
12 |  LS' ⇐ avaliaInstâncias(C_k, TS, maxDisag) ;   /* árvores (do comitê)
   |  emitem opinião sobre as instâncias, as que dividiram a opinião
   |  do comitê são retornadas em LS' */
13 |  rotulo(LS', maxRotPorIt);
14 |  LS ⇐ LS + LS';
15 |  atualizaConf(WT, CVote, C_k);
16 |  NC ⇐ DT − C_k;          /* árvores que não são do comitê atual */
17 |  C4.5(NC, LS);                   /* treina as árvores de decisão */
18 |  avaliaInstâncias(NC, AS); ;  /* árvores (do comitê) emitem opinião
   |  sobre as instâncias */
19 |  WT ⇐ atualizaPesosArv(DT, AllVote, C_k);
20 |  k + +;
21 até LS' = ∅;
   ;  /* repete até o comitê não ter dúvida sobre nenhuma instância */
22 R ⇐ avaliaInstâncias(C_k, TS);
```

a características), isso porque a geração de múltiplos comitês para avaliar as instâncias possibilita a identificação de diversos padrões. O *ALMa-CC* tem a vantagem de ter um critério de parada próprio, independente da quantidade de iterações, enquanto que o *ALMa-OC* executa por um número definido de iterações estabelecido experimentalmente.

Algoritmo 6: atualizaConf

Entrada: $WT, CVote, C_k$
Saída: WC

```
1  para cada AS_i faça
2      WC_i ⇐ 0;
3      se AS_i ∈ LS então
           ;          /* se a instância foi rotulada pelo especialista */
4          se AS_i.rotulo == VERDADEIRO então
5              WC_i ⇐ 1;
6          senão
7              WC_i ⇐ −1;
8          fim se
9      senão
10         para cada DT_t ∈ C_k faça
11             se CVote_{t,i} == VERDADEIRO então
12                 WC_i ⇐ WC_i + WT_t;
13             senão
14                 WC_i ⇐ WC_i − WT_t;
15             fim se
16         fim para cada
17     fim se
18 fim para cada
19 normaliza(WC_i, −1, 1);
```

	ALMa-OC	*ALMa-CC*
Características	Instâncias identificadas positivas pelo comitê atual são retornadas como resposta. Método executa por um determinado número de iterações	Instâncias identificadas positivas pelo comitê atual são utilizadas para avaliar a população de árvores. Método executa até que o comitê não tenha divergência de opiniões sobre as instâncias
Prós	Identifica diferentes padrões de instâncias positivas. Menor intervenção com o usuário	Método tem autonomia para finalizar o processo. Menos iterações para finalizar
Contras	Aprendizado não-monotônico Falta de critério de parada.	Apenas um comitê avalia as instâncias e dá o rótulo final Muitas interações com o usuário. Quantidade de interações deve ser controlada.

Tabela 3.4: Diferenças entre as duas abordagens de aprendizado ativo.

Capítulo 4

Experimentos

Nesta seção apresentamos os experimentos realizados para validação dos métodos descritos no Capítulo 3. Primeiro, fazemos uma caracterização das tarefas de *matching* que foram utilizadas para avaliar os métodos. Em seguida, mostramos a configuração utilizada nos métodos propostos. Depois, apresentamos os resultados alcançados pelo baseline e pelos métodos propostos para as tarefas de *matching* relacionadas à Ordem de Compra que foram usados na avaliação do *baseline* descrito em [Do and Rahm, 2002]. Em seguida, apresentamos os resultados alcançados pelos métodos para as tarefas de *matching* baseadas em ontologias da *Ontology Alignment Evaluation Initiative*[1].

4.1 Caracterização das tarefas de *matching*

Para realização dos experimentos, dois conjuntos de testes foram utilizados.

O primeiro conjunto (*Dataset*-1) é formado por cinco esquemas XDR relacionados à ordem de compra disponíveis na página do grupo de Banco de Dados da Universidade de Leipzig[2], foram definidas dez tarefas de *matching* pareando estes esquemas. Algumas características desse conjunto estão resumidas na Tabela 4.1. Maiores detalhes podem ser encontrados em [Do and Rahm, 2002]. Esse conjunto de esquemas foi escolhido por ser o mesmo utilizado na avaliação do *baseline*.

Esquema	CIDX	Excel	Noris	Paragon	Apertum
Profundidade	4	4	4	6	5
#Caminhos	40	54	65	80	147
#Folhas	33	42	54	68	118
#Nós Internos	7	12	11	12	29

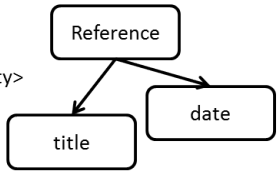
Tabela 4.1: Resumo de características do *Dataset*-1

O segundo conjunto (*Dataset*-2) compreende uma tarefa de *matching* onde os es-

[1] http://oaei.ontologymatching.org/

[2] http://dbs.uni-leipzig.de/bdschemamatching

```
<owl:Class rdf:ID="Reference">
  <rdfs:label xml:lang="en">Reference</rdfs:label>
  <rdfs:comment xml:lang="en">Base class for all entries</rdfs:comment>
  <rdfs:subClassOf>
   <owl:Restriction>
            <owl:onProperty rdf:resource="#date" />
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
   </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
   <owl:Restriction>
            <owl:onProperty rdf:resource="#title" />
            <owl:maxCardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:maxCardinality>
...
```




```
  ...
  <owl:DatatypeProperty rdf:ID="title">
    <rdfs:domain rdf:resource="#Reference" />
    <rdfs:range rdf:resource="&xsd;string" />
    <rdfs:label xml:lang="en">title</rdfs:label>
    <rdfs:comment xml:lang="en">The title of the
  work.</rdfs:comment>
   </owl:DatatypeProperty>
  ...
```

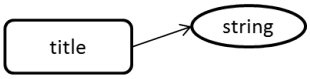


Figura 4.1: Transformação da ontologia RDF na representação utilizada pelo *ALMa*.

quemas são relacionados a tipos de vinhos disponíveis na página da *Information Interpretation and Integration Conference*[3] e onze tarefas de *matching* relacionadas a referências bibliográficas encontradas na página da *EON Ontology Alignment Contest*[4]. Esse conjunto foi escolhido por apresentar diferentes desafios como esquemas em linguas diferentes e esquemas sem correspondências válidas.

Note que esses esquemas foram originalmente utilizados no problema de alinhamento de ontologias e não para o problema de casamento de esquemas. Assim, para a sua utlização neste experimento, convertemos a representação semanticamente equivalente utilizando o modelo ER.

Para isso algumas propriedades das ontologias RDF foram suprimidas. Considere o exemplo da Figura 4.1, as cardinalidades das propriedades de uma classe foram suprimidas devido a não-utilização desse tipo de informação por nenhum dos *matchers* presentes na biblioteca apresentada na Seção 2.4.1, o tipo de dado de um elemento é obtido durante o processo de tranformação, pois, existe um *matcher* associado a este tipo de informação.

Os tipos de informação aproveitados das ontologias foram nomes dos elementos, tipos de dados e a estrutura hierárquica entre eles. Todas as outras informações disponíveis como cardinalidades e intervalos de dados foram descartadas, pois, nenhum dos *matchers* utilizados neste trabalho utiliza esse tipo de informação.

As características desse conjunto estão resumidas na Tabela 4.2, os esquemas re-

[3]http://www.atl.external.lmco.com/projects/ontology/i3con.htmlPart
[4]http://oaei.ontologymatching.org/2004/Contest/

lacionados a vinho são representados por *Wine* e aqueles relacionados a referências bibliográficas são representados por RB (esquemas semelhantes foram agrupados).

Esquema	Wine-A-B	RB-101-103-104	RB-102	RB-201-202-204-205	RB-206	RB-221	RB-222
Profundidade	3	4	6	4	5	3	3
#Caminhos	33	71	45	71	72	105	99
#Folhas	30	56	24	56	57	82	82
#Nós Internos	3	15	21	15	15	23	17

Tabela 4.2: Resumo de características do *Dataset*-2

Ao todo, vinte e duas tarefas de *matching* formam o conjunto de avaliação dos métodos. Os índices das tarefas, esquemas envolvidos e características específicas (se houver) são mostrados na Tabela 4.3. No *Dataset*-2, os esquemas foram casados somente com o esquema *RefBib101*, pois, o gabarito com as correspondências corretas para estas tarefas foi disponibilizado junto com as ontologias.

4.2 Configurações

O *baseline* (COMA) [Do and Rahm, 2002] apresenta um conjunto de doze *matchers* sendo sete simples e cinco híbridos. Segundo os autores, o COMA alcança melhores resultados utilizando os cinco *matchers* híbridos combinados através da média aritmética. Esta versão do COMA com os cinco *matchers* selecionados será referenciada por COMA-SEL5 e a versão com todos os *matchers* será o COMA-ALL. Vale lembrar que os *matchers* utilizados por COMA-SEL5 foram os cinco *matchers* que, combinados, alcançavam melhores resultados nesse *dataset*.

Os métodos propostos utilizam os dados gerados pelos doze *matchers* da biblioteca do COMA. Os métodos serão referenciados durante este Capítulo como *ALMa-OC* e *ALMa-CC*.

As instâncias utilizadas pelos algoritmos de aprendizado são representadas pelos pares de elementos dos esquemas e os valores de similaridade atribuídos pelos *matchers* como vistos na Tabela 3.1.

Nas seções seguintes apresentamos a parametrização para cada um dos métodos e experimentos relacionados.

4.2.1 *ALMa-OC*

O *ALMa-OC* utiliza uma população de árvores de decisão. Destas, apenas uma parcela (a cada iteração) faz parte de um comitê e tem o poder de decidir sobre a classe de uma instância. Experimentos foram realizados para definir a quantidade de árvores

utilizadas. Foram testadas versões do *ALMa-OC* com 20, 30 e 40 árvores na população. Cada tarefa foi executada 15 vezes. Os gráficos de precisão, revocação e média-F para cada tarefa deste experimento são mostrados no Apêndice A.1. O resultado médio desse experimento é mostrado na Tabela 4.4.

Sendo assim, a população então foi definida em 30 árvores sendo que, na primeira iteração, 12 árvores são formadas por apenas um *matcher* (uma árvore para cada *matcher*) e as árvores restantes utilizam no mínimo 3 *matchers* no treinamento. O número mínimo de elementos do comitê foi definido para 5 árvores, geralmente, o comitê envolve mais elementos, pois, várias árvores obtêm o mesmo valor de confiança.

O aprendizado do comitê é altamente dependente do conjunto inicial de instâncias escolhidas. Instâncias iniciais com características semelhantes, ou seja, com valores semelhantes dados pelos *matchers*, levam a comitês que convergem rapidamente e param de descobrir instâncias verdadeiras logo nas primeiras iterações. Instâncias iniciais espalhadas levam a comitês que dividem opiniões e utilizam mais iterações para descobrir instâncias verdadeiras e por vezes ficam algumas iterações sem classificar nenhuma instância como sendo da classe `VERDADEIRO`.

Na Figura 4.2 ilustramos esse comportamento através de um experimento realizado com a tarefa 1. Nota-se que na Figura 4.2(a) a escolha de instâncias iniciais semelhantes levou a uma convergência contínua de forma mais rápida. A situação inversa ocorre na Figura 4.2(b) onde instâncias de características diferentes foram utilizadas para formar o conjunto inicial.

Para solucionar esse problema, as instâncias são agrupadas utilizando o algoritmo *KMeans* sendo a similaridade entre instâncias o fator para a definição dos *clusters*. A medida de distância utilizada foi a Distância Euclidiana. A partir desse ponto, o conjunto inicial de instâncias rotuladas é definido por uma amostra aleatória dos *clusters* encontrados pelo algoritmo de agrupamento (uma instância de cada um dos *clusters* encontrados), o objetivo é obter instâncias de *clusters* que apresentem instâncias com características distintas.

O número de iterações escolhido foi 10. A partir desse ponto, poucas instâncias verdadeiras são descobertas através do desacordo do comitê e através do julgamento do comitê, além de manter a percentagem de instâncias utilizada para aprendizado baixa.

Em cada iteração, cinco instâncias são rotuladas pelo especialista/oráculo. Uma instância só recebe o rótulo `VERDADEIRO` pelo comitê quando este a julga verdadeira com uma confiança de valor mínimo 0.70. Essa confiança representa uma medida de certeza do rótulo dado pelo comitê. Este valor foi definido empiricamente utilizando três valores de confiança (C = {0.5, 0.6, 0.7}). Os gráficos de precisão, revocação e média-F para cada tarefa deste experimento são mostrados no Apêndice A.2. O resultado médio desse experimento é mostrado na Tabela 4.5.

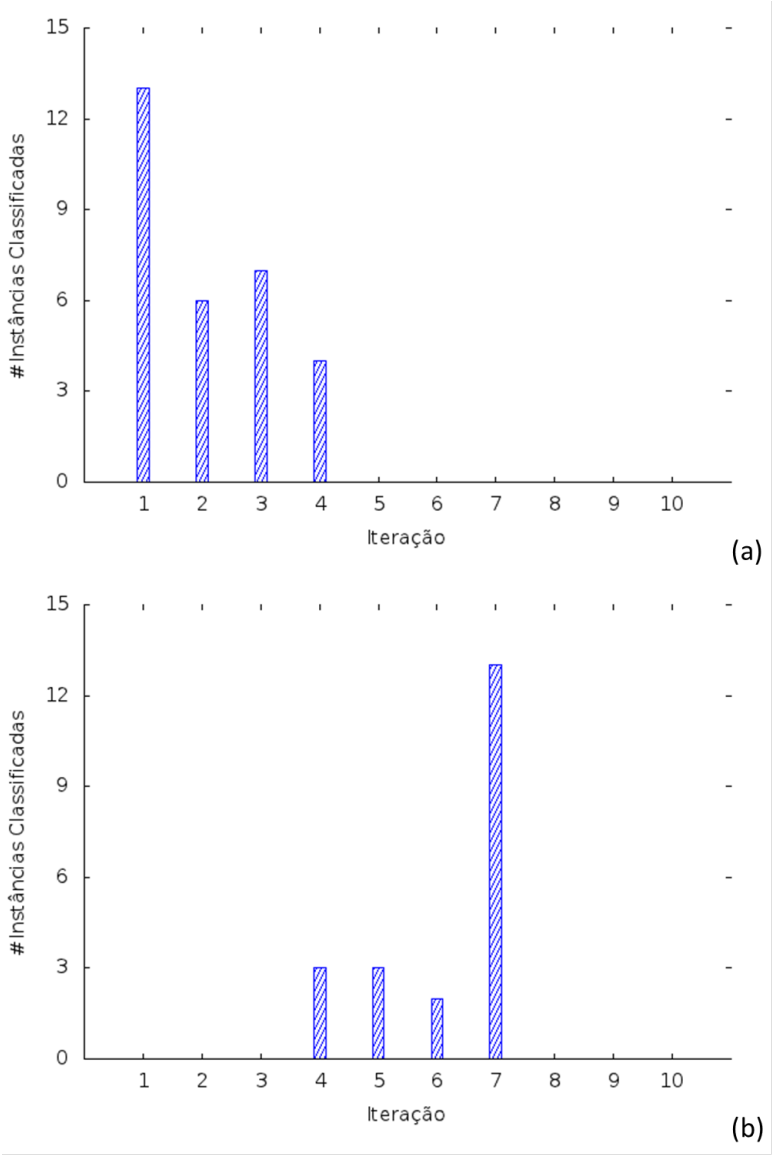


Figura 4.2: Quantidade de instâncias descobertas pelo comitê por iteração.

A configuração utilizada pelo *ALMa-OC*é resumida na Tabela 4.6.

4.2.2 *ALMa-CC*

Assim como no *ALMa-OC*, foi realizado um experimento para definir a quantidade de árvores que comporiam a população do método. Versões com 20, 40 e 60 árvores foram testadas. Os gráficos de precisão, revocação e média-F para cada tarefa deste experimento são mostrados no Apêndice A.3. O resultado médio desse experimento é mostrado na Tabela 4.7.

Semelhante ao *ALMa-OC*, 12 árvores são formadas por apenas um *matcher* na primeira iteração (uma árvore para cada *matcher*) e as restantes utilizam combinações de *matchers*. Para que o número de árvores com apenas um *matcher* não seja mais da metade do total de árvores, foi utilizada uma população de 40 árvores. O comitê deve ser formado por, no mínimo, 10 árvores. Diferente do *ALMa-OC*, somente o último comitê avalia as instâncias. Sendo assim, as iterações com o especialista servem para obter instâncias que mais geram discordância entre os elementos do comitê, esse processo é realizado até a convergência do comitê.

Em cada iteração, no máximo 15 instâncias são rotuladas pelo especialista (comitês iniciais tendem a discordar em muitas instâncias). O comitê atinge um estado de `dúvida` sobre uma instância se o índice de discordância ultrapassa 15%. O conjunto inicial de instâncias rotuladas é definido por uma amostra aleatória dos *clusters* encontrados pelo algoritmo de agrupamento *KMeans* (uma instância de cada um dos 15 *clusters* encontrados) e 13 instâncias do conjunto *TOP* e 13 instâncias do conjunto *BOT*. A quantidade de instâncias no primeiro treino para o *ALMa-CC* é consideravelmente maior que a do *ALMa-OC* pois, deseja-se evitar que os primeiros comitês sejam formados por árvores semelhantes.

A parametrização do *ALMa-CC* está descrita na Tabela 4.8.

4.3 Avaliação

As métricas utilizadas para avaliar os métodos são os já bastante usados em Recuperação de Informação: precisão e revocação. As métricas procuram avaliar a eficiência dos métodos, a precisão mede o quanto as respostas retornadas por um método estão corretas e a revocação mede o quanto de respostas corretas do total de respostas corretas foi retornado.

A precisão e a revocação não conseguem, sozinhas, indicar a boa qualidade de um método, então, diversas propostas de métricas combinando as duas anteriores são apresentadas na literatura como a média harmônica *F* [Baeza-Yates and Ribeiro-Neto, 2011]. Apenas para validação da implementação do

baseline, utilizamos a métrica *Overall* [Melnik et al., 2002], que procura mensurar o esforço do usuário para inserir casamentos não encontrados e retirar casamentos incorretos retornados na resposta e foi utilizada na publicação do *baseline*.

Sejam R um conjunto de respostas dadas por um método e R_C o subconjunto de R que contém as respostas corretas. Seja também, T o conjunto de todas as respostas corretas pré-definidas por um gabarito.

A precisão é então definida como:

$$Prec = \frac{|R_C|}{|R|}$$

A revocação é definida por:

$$Rev = \frac{|R_C|}{|T|}$$

A média harmônica F é definida por:

$$MediaF = \frac{2 * Prec * Rev}{Prec + Rev}$$

4.4 Resultados

Nesta seção apresentamos os resultados dos experimentos realizados.

Para efeitos de comparação, foram executadas duas versões do COMA (ambas implementação própria). A primeira – COMA-ALL – utiliza todos os *matchers* disponíveis, a segunda – COMA-SEL5 – utiliza apenas os 5 *matchers* híbridos que, segundo os autores, alcança melhores resultados.

Como o *ALMa-OC* e o *ALMa-CC* utilizam um componente gerador de números aleatórios (para selecionar instâncias dos *clusters* gerados pelo *KMeans* e para escolher os *matchers* para cada árvore de decisão), cada tarefa de casamento foi executada 30 vezes, os resultados reportados para uma tarefa correspondem a média das execuções.

Os métodos foram implementados em JAVA e executam em um computador com um processador Intel i3, 4GB de memória RAM e um sistema operacional de 64 bits.

Nas seções seguintes, são detalhados os resultados para cada *dataset*, são reportados os dados sobre instâncias de aprendizado e são feitas as considerações.

4.4.1 *Dataset*-1

Os resultados de precisão, revocação e média-F para os métodos são mostrados nas Figuras 4.3(a), Figuras 4.3(b) e Figuras 4.3(c) respectivamente.

Podemos observar que, de fato, a estratégia COMA-SEL5 obtém melhores resultados que a versão COMA-ALL. Isso se dá ao fato de que *matchers* simples tendem

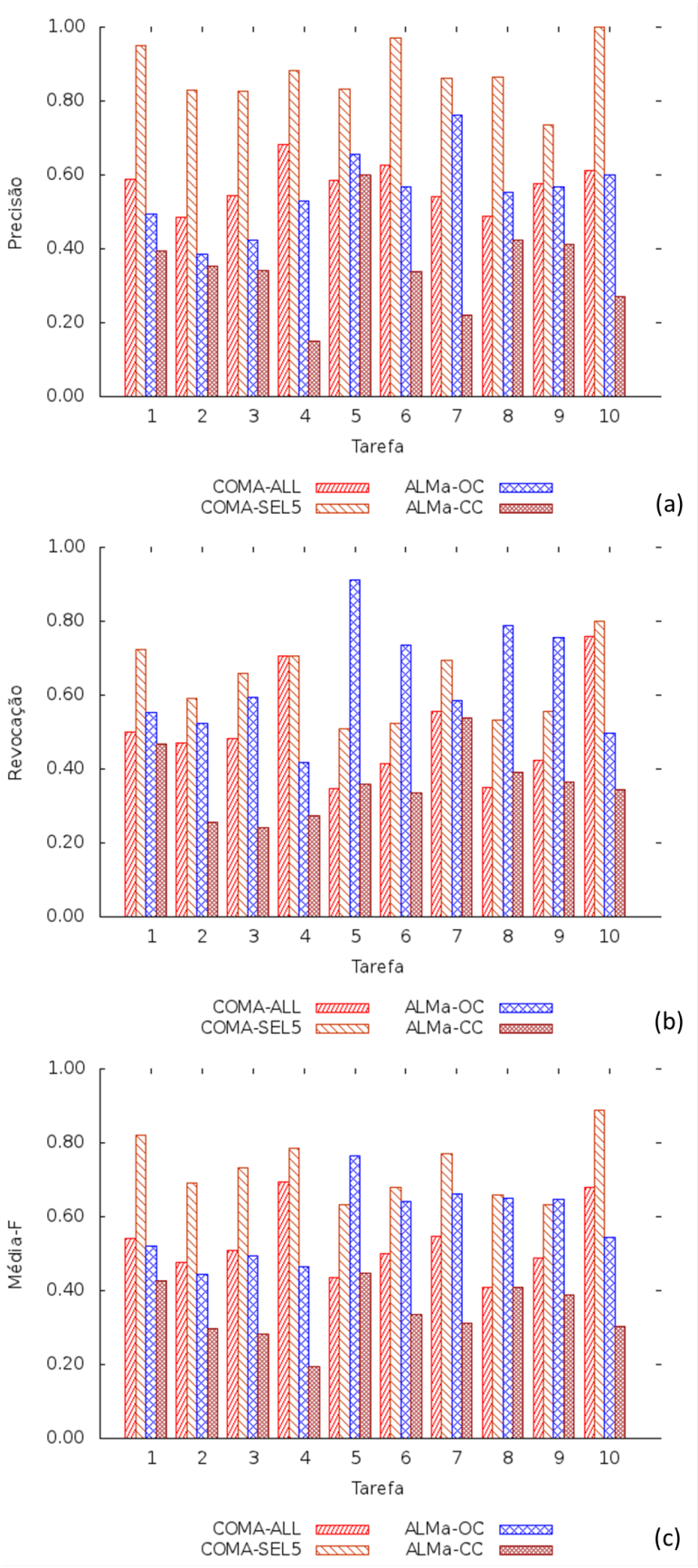


Figura 4.3: Resultados dos quatro métodos para o *Dataset*-1. (a) Precisão. (b) Revocação. (c) Média-F.

a ser enganados pois avaliam as instâncias somente por um aspecto (como nomes similares, por exemplo), enquanto que os *matchers* híbridos utilizados em COMA-SEL5 conseguem combinar aspectos como caminho e propriedades estruturais (nó interno ou folha, por exemplo).

Os *matchers* selecionados em COMA-SEL5 foram escolhidos manualmente por apresentar os melhores resultados [Do and Rahm, 2002] nesse conjunto de testes. O COMA-SEL5 obteve média-F igual a 0.72 enquanto que o COMA-ALL obteve média-F igual a 0.53.

O *ALMa-OC* obtém revocação semelhante ao COMA-SEL5, ou seja, o número de instâncias corretas retornadas é o mesmo comparado ao número total de instâncias corretas. A diferença na precisão se dá ao fato de que, o COMA-SEL5 utiliza uma heurística de seleção de casamentos corretos que funciona bem para este conjunto de instâncias, enquanto que as árvores de decisão presentes no *ALMa-OC* e no *ALMa-CC* não conseguem encontrar o ponto exato que separa as instâncias verdadeiras das instâncias falsas.

O resultado médio para as dez tarefas do Dataset-1 são mostrados na Tabela 4.9.

4.4.2 *Dataset*-2

Os resultados de precisão, revocação e média-F para os quatro métodos são mostrados nas Figuras 4.4(a), Figuras 4.4(b) e Figuras 4.4(c) respectivamente.

O *ALMa-OC* obteve maiores valores de revocação e média-F dentre as estratégias testadas (média-F = 0.70). Como os esquemas desse *dataset* não são "bem-comportados", pode-se observar que a estratégia fixa do COMA não responde bem a este tipo de tarefa. Isso se dá pois as instâncias verdadeiras não são aquelas em que os *matchers* atribuem valor próximo de 1, essas instâncias possuem valores menores não reconhecidos pelo método. A estratégia fixa do COMA de selecionar as instâncias com valores mais altos mostra-se precisa, porém não é abrangente. A utilização dos múltiplos comitês do *ALMa-OC* possibilita a identificação de diversos padrões de valores dados pelos *matchers*, aumentando assim a revocação do método.

O resultado médio para as dez tarefas do Dataset-2 são mostrados na Tabela 4.10.

4.4.3 Aprendizado

O aprendizado ativo consiste em deixar o método escolher quando utilizar instâncias para treino e então requisitá-las de um oráculo. O *ALMa-OC* e o *ALMa-CC* utilizam essa abordagem e tentam melhorar o desempenho de uma abordagem fixa obtendo algumas instâncias rotuladas. A Tabela 4.11 mostra a percentagem média de instâncias que foram utilizadas para treino durante os experimentos para cada tarefa de casamento.

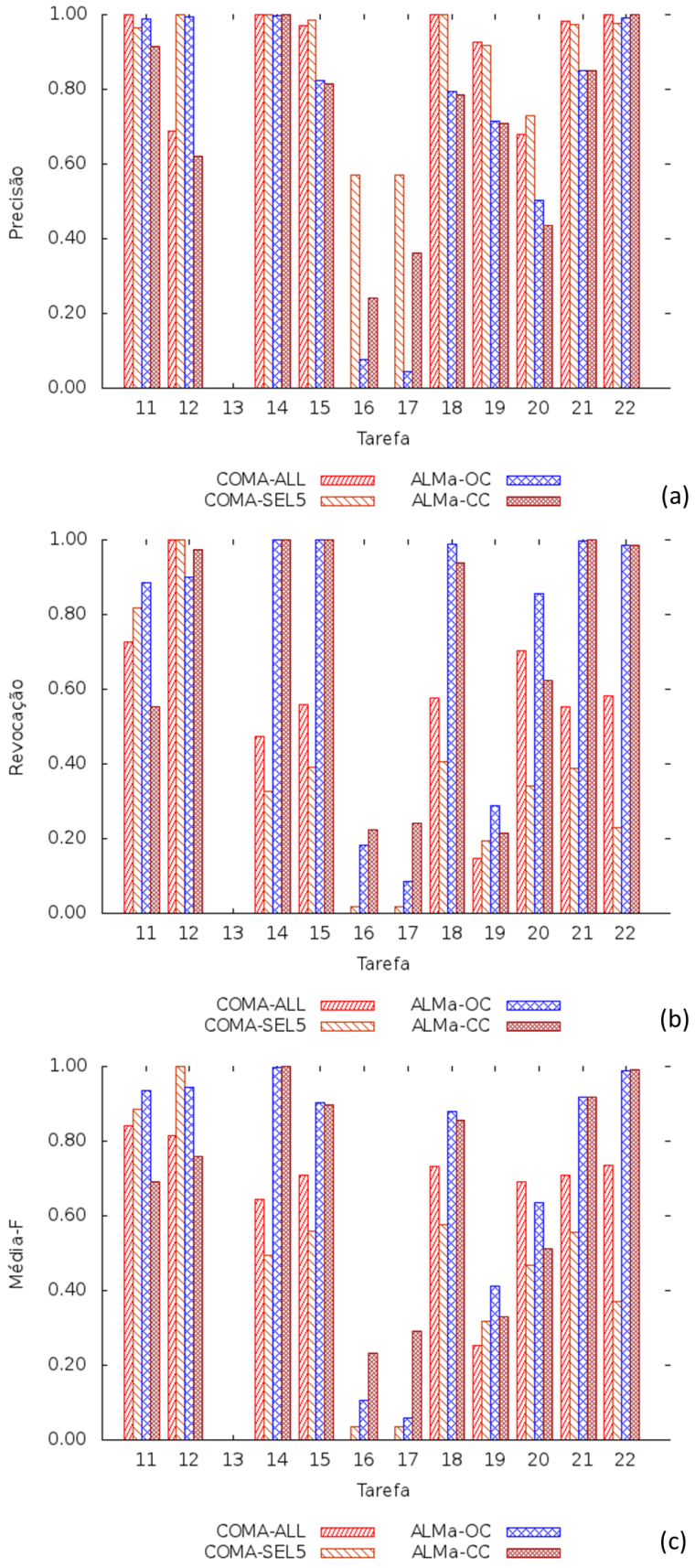


Figura 4.4: Resultados dos quatro métodos para o *Dataset*-2. (a) Precisão. (b) Revocação. (c) Média-F.

Também podemos observar que a quantidade de instâncias rotuladas pelo especialista para treino das árvores de decisão foi, na média, pouco mais de 1% do total de instâncias para o *ALMa-OC* e quase 2% para o *ALMa-CC*.

O primeiro conjunto de instâncias de treino têm um papel importante uma vez que instâncias com características diferentes são melhores para geração de dúvida pelo comitê e o aprendizado ser efetivo. Uma escolha ruim pode levar a um aprendizado incorreto ou a convergência rápida do comitê resultando na qualidade ruim das respostas. Para evitar este problema, utilizamos o algoritmo KMeans para dividir as instâncias em grupos e, uma instância de cada grupo gerado é escolhida para ser rotulada e fazer parte do conjunto de treino.

4.4.4 Considerações

Ao todo, vinte e duas tarefas de casamento foram submetidas aos métodos. Podemos observar uma pesquena diferença de média-F do COMA-SEL5 e do *ALMa-OC* embora tenham obtido diferentes resultados de precisão e revocação. As estratégias COMA-ALL e *ALMa-CC* obtiveram resultados inferiores.

O COMA-ALL apresentou desempenho semelhante nos dois *datasets* quanto à média-F. O COMA-SEL5 mostrou-se eficiente para o *Dataset*-1 (conjunto em que foi originalmente publicado), a estratégia fixa e os *matchers* especificamente escolhidos alcançaram valor médio de *Overall* igual a 0.54. Este experimento também serviu para validar a implementação do método. A publicação original reporta que o valor de *Overall* médio alcançado foi 0.6 [Do and Rahm, 2002].

No segundo *dataset*, o COMA-ALL desempenhou-se como esperado para as tarefas 11 e 12. Essas tarefas não apresentaram desafios à estratégia estática pois utilizam esquemas semelhantes (nomes e estruturas). Para as outras tarefas, o método obteve bons resultados de precisão derivados de um critério de escolha que prioriza instâncias com altos valores dados por todos os *matchers*, mas não conseguiu uma abrangência e obteve resultados baixos de revocação.

O *ALMa-OC* mostrou-se não tão eficiente no *Dataset*-1 onde as instâncias positivas e negativas possuem valores muito próximos. A avaliação por múltiplos comitês consegue encontrar instâncias com características diferentes (possíveis *outliers*) e a capacidade de generalização das árvores levou a valores mais altos de revocação. Ao contrário do *ALMa-CC* que ao utilizar apenas um comitê final, alcança menos instâncias corretas dentre o conjunto total de instâncias corretas levando a um menor índice de revocação e, consequentemente, média-F.

Os resultados médios para os métodos testados estão disponíveis na Tabela 4.12.

Dataset	Índice	Esquemas	Caracteristica	#Instâncias
1	1	Apertum-CIDX		5880
1	2	Apertum-Excel		7938
1	3	Apertum-Noris		9555
1	4	Apertum-Paragon		11760
1	5	CIDX-Excel		2160
1	6	CIDX-Noris		2600
1	7	CIDX-Paragon		3200
1	8	Excel-Noris		3510
1	9	Paragon-Excel		4320
1	10	Paragon-Noris		5200
2	11	WineA-WineB	Idiomas diferentes	1089
2	12	RefBib101-RefBib101	Mesmo esquema	5041
2	13	RefBib101-RefBib102	Esquemas sem casamentos válidos	3195
2	14	RefBib101-RefBib103	Ambiguidades presentes nos esquemas	5041
2	15	RefBib101-RefBib104	Ambiguidades presentes nos esquemas	5041
2	16	RefBib101-RefBib201	Nomes de elementos trocados	5041
2	17	RefBib101-RefBib202	Nomes de elementos trocados	5041
2	18	RefBib101-RefBib204	Convenções e abreviaturas diferentes	5041
2	19	RefBib101-RefBib205	Uso de sinônimos	5041
2	20	RefBib101-RefBib206	Esquema em outro idioma	5112
2	21	RefBib101-RefBib221	Sem hierarquia	7455
2	22	RefBib101-RefBib222	Hierarquia reduzida	7029

Tabela 4.3: Tarefas de *matching* definidas

Qt.Árvores	20	30	40
Precisão	0,5371	0,5591	0,5227
Revocação	0,5125	0,6175	0,5623
Média-F	0,5280	0,5835	0,5340

Tabela 4.4: Resultado médio do teste de população de árvores para o *ALMa-OC*.

	C=0.5	C=0.6	C=0.7
Precisão	0.4511	0.4742	0.5457
Revocação	0.5818	0.5797	0.5750
Média-F	0.5064	0.5148	0.5576

Tabela 4.5: Resultado médio do teste de variação do índice de confiança para o *ALMa-OC*.

Configuração	Valor
Seleção de instâncias inicial ($kClusters$)	KMeans(5)
Qt. de árvores	30
Qt. mínima de *matchers* por árvore	3
Tamanho mínimo do comitê ($minTamC$)	5
Número de iterações ($maxIT$)	10
Qt. instâncias por iteração ($instPorIt$)	5
Confiança mínima ($minConf$)	0.70
Árvore com único *matcher*	sim

Tabela 4.6: Configurações do *ALMa-OC*.

Qt.Árvores	20	40	60
Precisão	0.6004	0.5204	0.5007
Revocação	0.4790	0.5103	0.5094
Média-F	0.5023	0.4975	0.4896

Tabela 4.7: Resultado médio do teste de população de árvores para o *ALMa-CC*.

Configuração	Valor
Seleção de instâncias inicial ($kClusters, qtTB$)	KMeans(15) e Top/Bottom(13)
Qt. de árvores	40
Qt. mínima de *matchers* por árvore	5
Tamanho mínimo do comitê ($minTamC$)	10
Desacordo máximo ($maxDisag$)	15%
Qt. máxima de intervenções por iteração ($maxRotPorIt$)	15
Árvore com único *matcher*	sim

Tabela 4.8: Configurações do *ALMa-CC*.

	COMA-ALL	COMA-SEL5	*ALMa-OC*	*ALMa-CC*
Precisão	0.573	0.876	0.554	0.350
Revocação	0.501	0.629	0.636	0.357
Média-F	0.528	0.729	0.583	0.340

Tabela 4.9: Resultado médio para as tarefas do *Dataset*-1.

	COMA-ALL	COMA-SEL5	*ALMa-OC*	*ALMa-CC*
Precisão	0.749	0.880	0.707	0.636
Revocação	0.483	0.375	0.685	0.594
Média-F	0.557	0.481	0.705	0.614

Tabela 4.10: Resultado médio para as tarefas do *Dataset*-2.

Tarefa	ALMa-OC	#Instâncias	ALMa-CC	#Instâncias
1	0,85%	50	1,17%	69
2	0,63%	50	1,15%	91
3	0,52%	50	1,19%	113
4	0,43%	50	0,65%	77
5	2,31%	50	3,87%	84
6	1,92%	50	2,58%	67
7	1,56%	50	2,66%	85
8	1,42%	50	2,15%	76
9	1,16%	50	1,82%	79
10	0,96%	50	1,65%	86
11	4,59%	50	4,66%	51
12	0,99%	50	1,42%	71
13	1,56%	50	1,28%	41
14	0,99%	50	0,87%	44
15	0,99%	50	0,93%	47
16	0,99%	50	1,89%	95
17	0,99%	50	2,34%	118
18	0,99%	50	1,18%	59
19	0,99%	50	1,74%	88
20	0,98%	50	1,52%	78
21	0,67%	50	0,77%	58
22	0,71%	50	0,70%	49

Tabela 4.11: Média da percentagem de instâncias utilizadas para treino das árvores e número de instâncias utilizadas.

	COMA-ALL	COMA-SEL5	*ALMa-OC*	*ALMa-CC*
Precisão	0,6654	0,8783	0,6341	0,4860
Revocação	0,4919	0,4964	0,6593	0,4626
Média-F	0,5432	0,5995	0,6482	0,4936

Tabela 4.12: Resultado médio dos experimentos para os quatro métodos testados.

Capítulo 5

Considerações Finais

O casamento de esquemas é uma tarefa importante em diversas aplicações e soluções utilizadas se apóiam na intervenção humana. Diferente das abordagens heurísticas que têm uma estratégia estática e da abordagem com aprendizado que utiliza uma grande quantidade de dados rotulados, propomos o *ALMa*, uma abordagem com aprendizado ativo no intuito de utilizar a experiência do usuário apenas quando necessário.

Comparamos o *ALMa* com um *baseline* publicado, o COMA foi escolhido por ser relativamente simples e pelo artigo associado conter um nível de detalhamento suficiente para que a reprodução da implementação fosse fiel. Uma barreira encontrada foi a não disponibilização dos códigos-fonte ou até de protótipos derivados de métodos publicados, e para os protótipos disponibilizados, a interface continha uma caixa-preta dificultando a sua utilização neste trabalho.

Mostramos, através de experimentos, que a estratégia fixa do *baseline* é eficiente no conjunto de teste que foi proposta mas não é abrangente para outras tarefas de casamento. Propomos uma alternativa a essa abordagem utilizando aprendizado ativo e comitês de árvores de decisão, o método *ALMa-OC* apresentou os melhores resultados para o segundo conjunto de teste alcançando média-F igual a 0.64. E mostramos que a percentagem de instâncias utilizadas para o treino da população de árvores presente nos métodos é inferior a 2%.

Este trabalho pode ser utilizado para nortear outros trabalhos como:

Testes com outros protótipos, utilizando *matchers* mais complexos: diversos outros tipos de *matchers* não utilizados na biblioteca do COMA podem ser testados. Funções de outros protótipos que utilizam bases de conhecimento como o WordNet[1] e *matchers* baseados em estatísticas de Recuperação de Informação por exemplo.

Utilização de outros algoritmos de aprendizagem de máquina: as árvores de decisão são simples e são sensíveis a mudanças no conjunto de treino, mas ou-

[1] http://wordnet.princeton.edu/

tros modelos de aprendizado podem ser utilizados. Algoritmos como *Random Forests* e técnicas como *Boosting* e Programação Genética podem ser testados para melhorar a eficácia do método.

Divisão da tarefa da aprendizado: uma abordagem de divisão-e-conquista pode ser utilizada em oposição a considerar todo o conjunto de instâncias durante a realização da tarefa. Pode-se dividir o conjunto de instâncias (com o KMeans, por exemplo) e executar o método em cada subconjunto. Pode-se ainda tentar utilizar o aprendizado adquirido em uma subtarefa como entrada para os outros subconjuntos.

Aplicação para o problema de Alinhamento de Ontologias: com a *Web* Semântica, as ontologias têm atraído bastante atenção dos pesquisadores e, semelhante ao problema de casamento de esquemas, tem bastantes aplicações na área de Integração de Dados.

Apêndice A

Detalhamento dos experimentos

Este apêndice apresenta alguns dados sobre os experimentos que não foram incluídos no capítulo original.

A.1 *ALMa-OC*: população de árvores

Este experimento foi realizado para verificar a influência da quantidade de árvores de decisão presentes na população. Os gráficos apresentados ilustram o valor médio de 15 execuções para cada tarefa, são mostradas a precisão, a revocação e a média-F para o *Dataset*-1(Figura A.1) e para o *Dataset*-2(Figura A.2).

Para os dois *datasets* a execução com 30 árvores apresentou melhores resultados de média-F.

A.2 *ALMa-OC*: confiança do comitê

Para atribuir definitivamente um rótulo positivo a uma instância, o comitê deve concordar que o rótulo é positivo com um valor mínimo de confiança. Este valor foi definido através deste experimento. Para cada tarefa de casamento, 15 execuções foram feitas e os resultados médios são reportados nos gráficos de precisão (Figura A.3(a) e Figura A.4(a)), revocação (Figura A.3(b) e Figura A.4(b)) e média-F (Figura A.3(c) e Figura A.4(c)) para o *Dataset*-1 e *Dataset*-2 respectivamente.

Os gráficos mostram que o aumento da certeza do comitê aumentou o grau de precisão do método e a revocação não foi prejudicada. Assim, a configuração com a confiança igual a 0.7 foi escolhida.

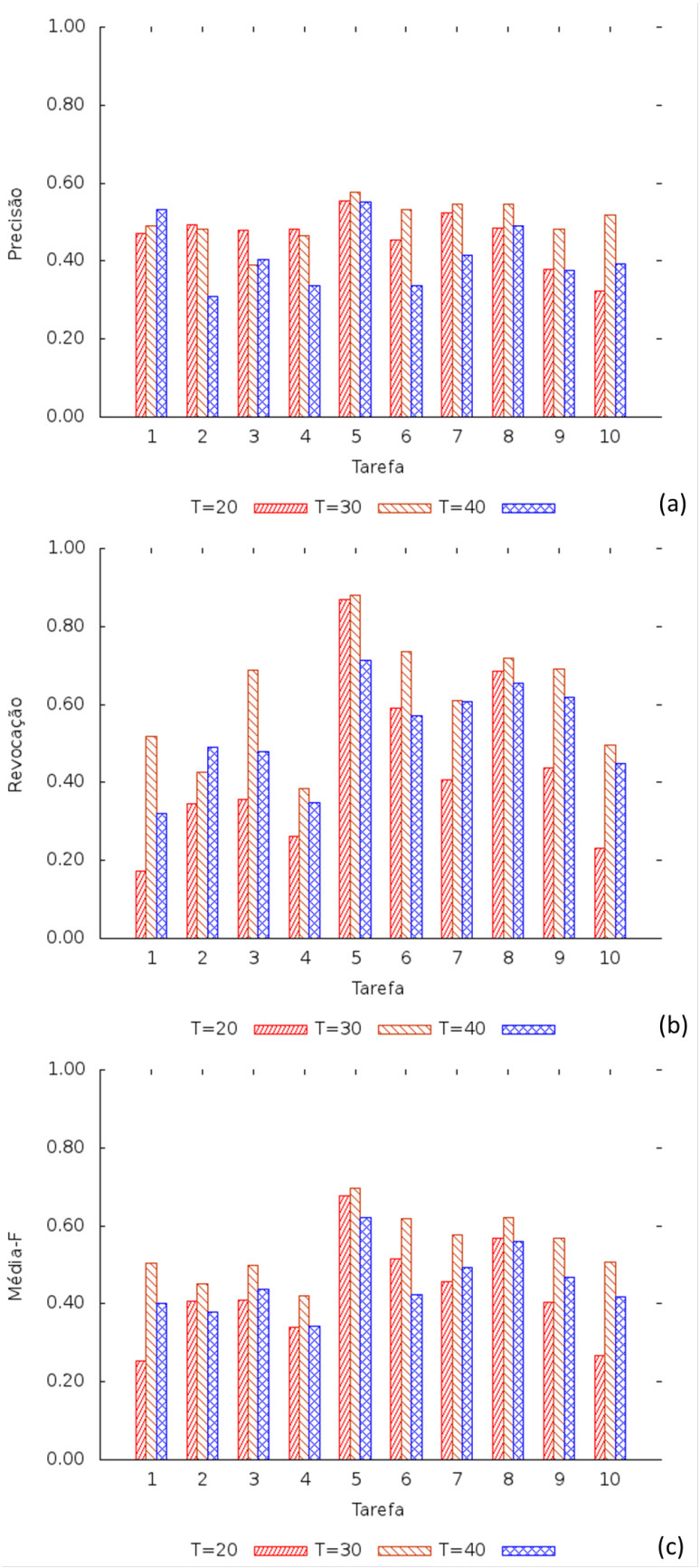


Figura A.1: Teste de população de árvores para o *ALMa-OC* (*Dataset*-1). (a) Precisão. (b) Revocação. (c) Média-F.

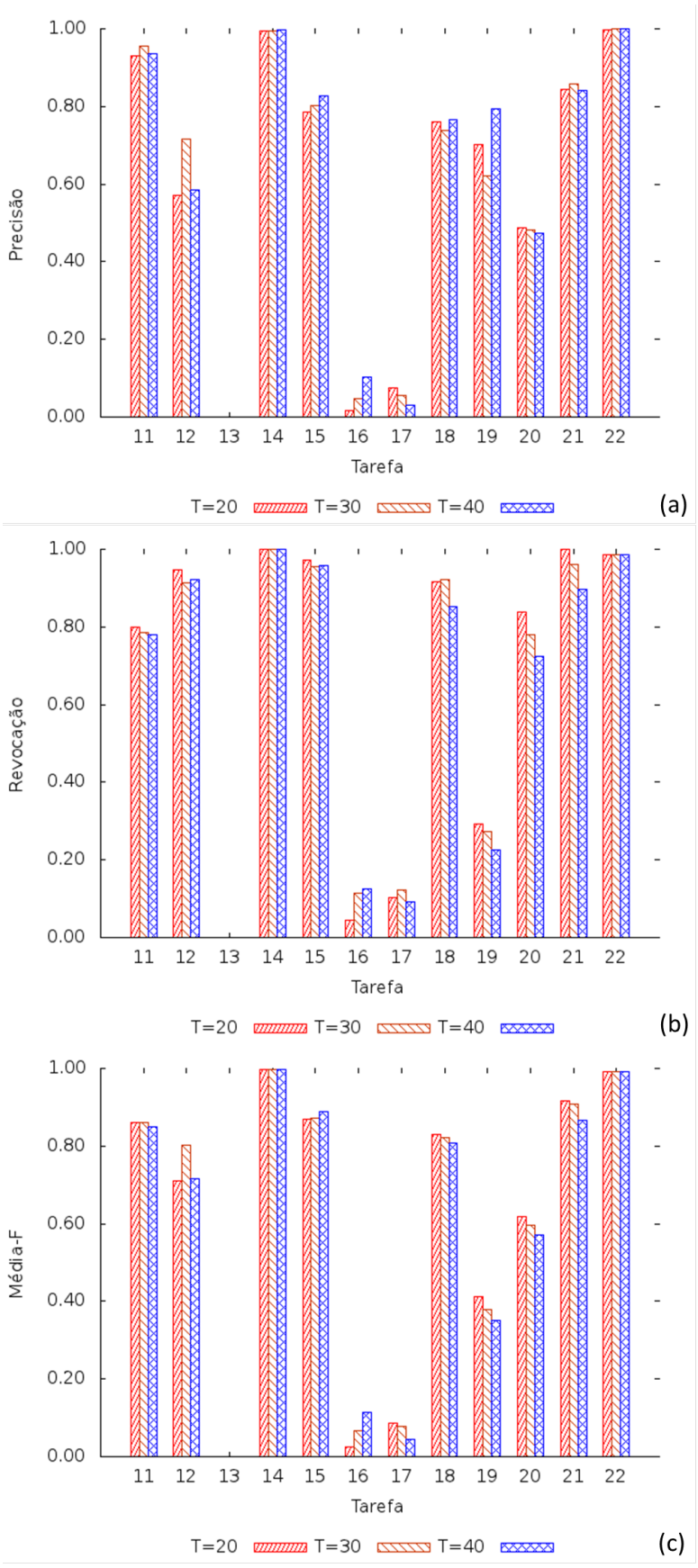


Figura A.2: Teste de população de árvores para o *ALMa-OC* (*Dataset*-2). (a) Precisão. (b) Revocação. (c) Média-F.

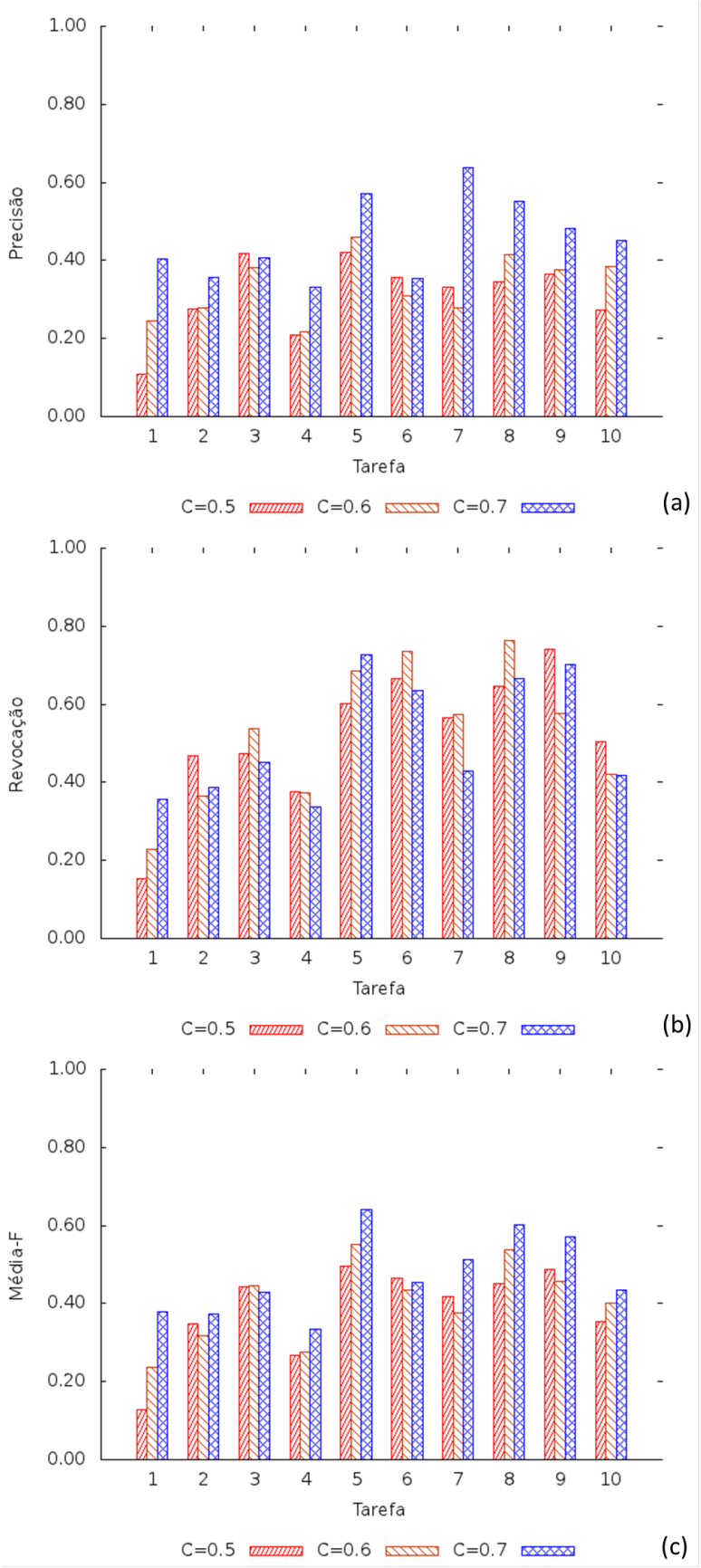


Figura A.3: Teste de confiança mínima para o *ALMa-OC* (*Dataset*-1). (a) Precisão. (b) Revocação. (c) Média-F.

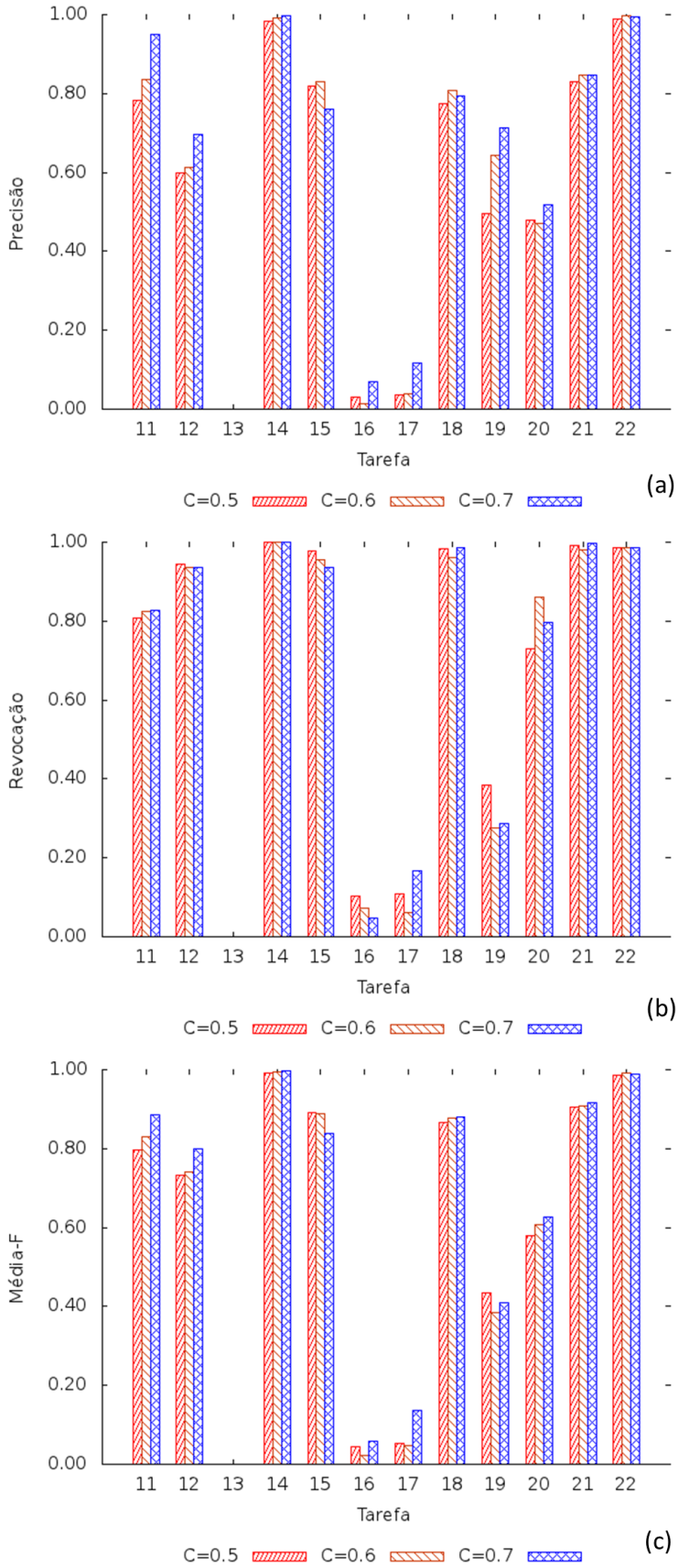


Figura A.4: Teste de confiança mínima para o *ALMa-OC* (*Dataset*-2). (a) Precisão. (b) Revocação. (c) Média-F.

A.3 *ALMa-CC*: população de árvores

Similar ao experimento com o *ALMa-OC*, também foram feitos experimentos para verificar a influência da quantidade de árvores na população do método. Os gráficos apresentados nas Figuras A.5 e Figura A.6 mostram o valor médio de precisão, revocação e média-F de 15 execuções para 20, 40 e 60 árvores para cada um dos *datasets*.

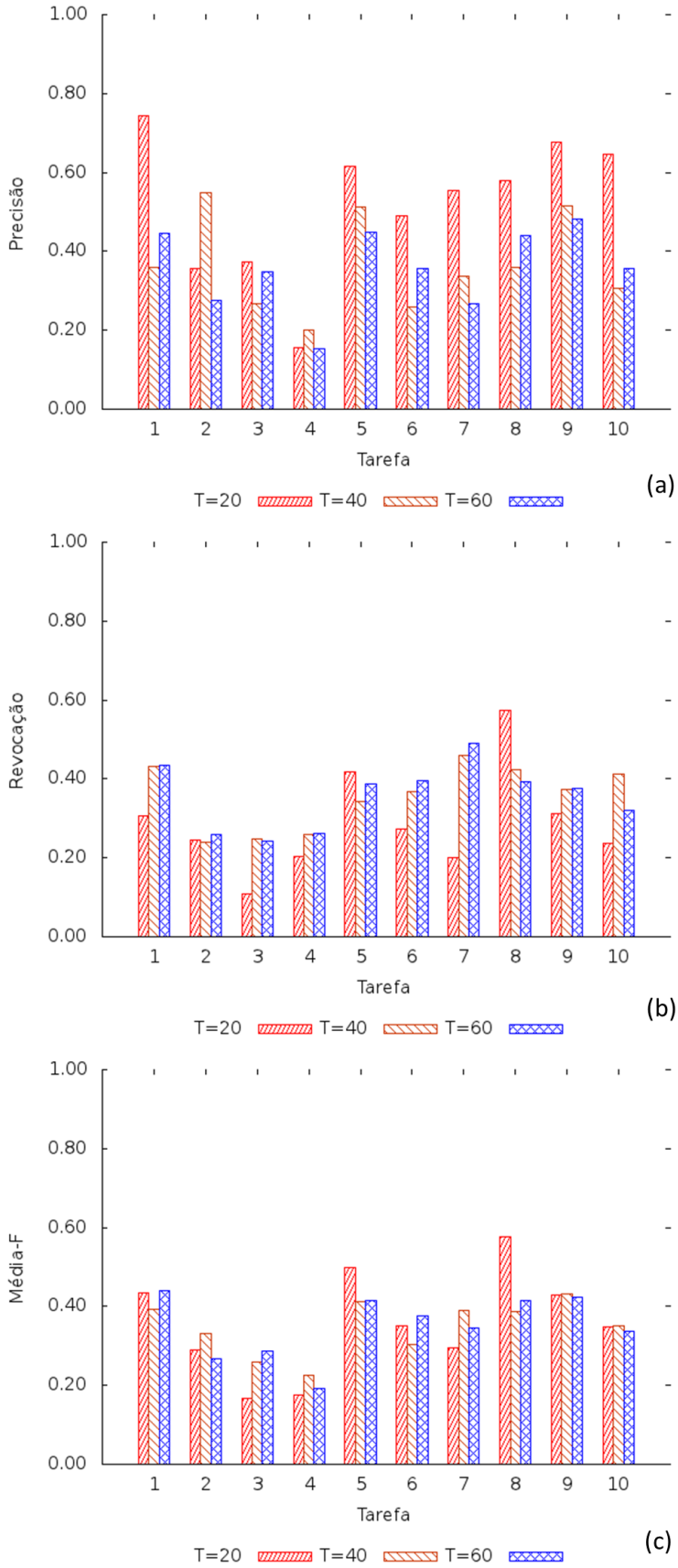


Figura A.5: Teste de população de árvores para o *ALMa-CC* (*Dataset*-1). (a) Precisão. (b) Revocação. (c) Média-F.

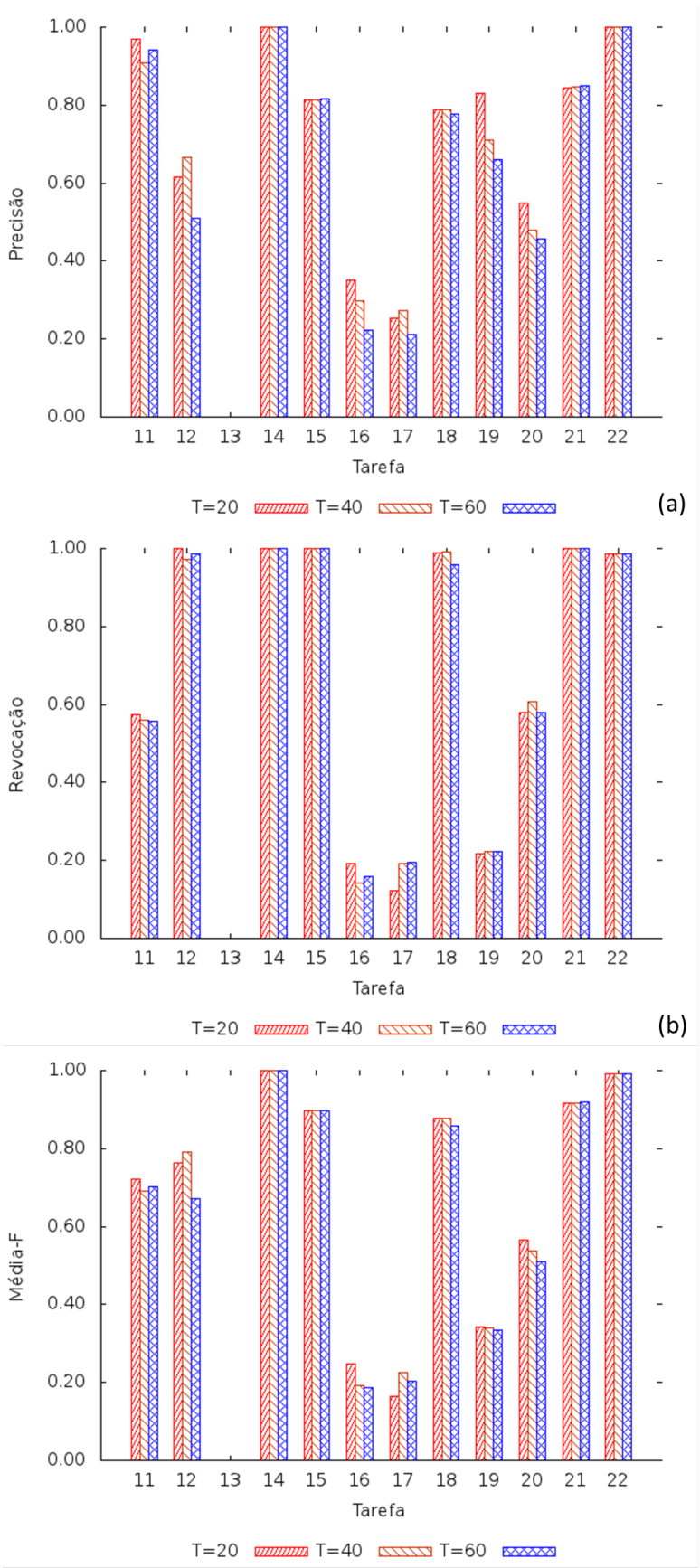


Figura A.6: Teste de população de árvores para o *ALMa-CC* (*Dataset*-2). (a) Precisão. (b) Revocação. (c) Média-F.

Referências Bibliográficas

[Aumueller et al., 2005] Aumueller, D., Do, H.-H., Massmann, S., and Rahm, E. (2005). Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 906–908, New York, NY, USA. ACM.

[Baeza-Yates and Ribeiro-Neto, 2011] Baeza-Yates, R. A. and Ribeiro-Neto, B. A. (2011). *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England.

[Ballard et al., 1998] Ballard, C., Herreman, D., Schau, D., Bell, R., Kim, E., and Valencic, A. (1998). *Data modeling techniques for data warehousing.* IBM Corp., Riverton, NJ, USA.

[Bellahsene et al., 2011] Bellahsene, Z., Bonifati, A., and Rahm, E., editors (2011). *Schema Matching and Mapping.* Springer.

[Beneventano et al., 2000] Beneventano, D., Bergamaschi, S., Castano, S., Corni, A., Guidetti, R., Malvezzi, G., Melchiori, M., and Vincini, M. (2000). Information integration: The momis project demonstration. In *Proceedings of the 26th International Conference on Very Large Data Bases*, VLDB '00, pages 611–614, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Bernstein and Melnik, 2004] Bernstein, P. A. and Melnik, S. (2004). Meta data management. In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, page 875.

[Bernstein et al., 2006] Bernstein, P. A., Melnik, S., and Churchill, J. E. (2006). Incremental schema matching. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 1167–1170. VLDB Endowment.

[Beygelzimer et al., 2009] Beygelzimer, A., Dasgupta, S., and Langford, J. (2009). Importance weighted active learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 49–56, New York, NY, USA. ACM.

[Bilke and Naumann, 2005] Bilke, A. and Naumann, F. (2005). Schema matching using duplicates. In *Data Engineering, 2005. Proceedings. 21st International Conference on*, ICDE '05, pages 69 – 80.

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., Secaucus, NJ, USA.

[Bonifati and Velegrakis, 2011] Bonifati, A. and Velegrakis, Y. (2011). Schema matching and mapping: from usage to evaluation. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 527–529, New York, NY, USA. ACM.

[Chukmol et al., 2005] Chukmol, U., Rifaieh, R., and Benharkat, N. (2005). Exsmal: Edi/xml semi-automatic schema matching algorithm. In *E-Commerce Technology, 2005. CEC 2005. Seventh IEEE International Conference on*, CEC'05, pages 422–425.

[Cohn et al., 1994] Cohn, D., Atlas, L., and Ladner, R. (1994). Improving generalization with active learning. *Machine Learning*, 15(2):201–221.

[Conrad et al., 1997] Conrad, S., Hoding, M., Saake, G., Schmitt, I., and Turker, C. (1997). Schema integration with integrity constraints. In *Advances in Databases, 15th British National Conf. on Databases*, BNCOD '15, pages 200–214. Springer-Verlag.

[Cruz et al., 2009] Cruz, I. F., Antonelli, F. P., and Stroe, C. (2009). Agreementmaker: efficient matching for large real-world schemas and ontologies. *Proc. VLDB Endowment*, 2(2):1586–1589.

[de Freitas et al., 2010] de Freitas, J., Pappa, G., da Silva, A., Gonçalves, M., Moura, E., Veloso, A., Laender, A., and de Carvalho, M. (2010). Active learning genetic programming for record deduplication. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, CEC '10, pages 1–8.

[Do and Rahm, 2002] Do, H.-H. and Rahm, E. (2002). Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 610–621. VLDB Endowment.

[Doan et al., 2000] Doan, A., Domingos, P., and Levy, A. Y. (2000). Learning source description for data integration. In *WebDB (Informal Proceedings)*, pages 81–86.

[Duchateau et al., 2009a] Duchateau, F., Coletta, R., Bellahsene, Z., and Miller, R. J. (2009a). (not) yet another matcher. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1537–1540, New York, NY, USA. ACM.

[Duchateau et al., 2009b] Duchateau, F., Coletta, R., Bellahsene, Z., and Miller, R. J. (2009b). Yam: a schema matcher factory. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2079–2080, New York, NY, USA. ACM.

[Gal, 2006] Gal, A. (2006). Why is schema matching tough and what can we do about it? *SIGMOD Rec.*, 35(4):2–5.

[Jian et al., 2005] Jian, N., Hu, W., Cheng, G., and Qu, Y. (2005). Falcon-ao: Aligning ontologies with falcon. pages 85–91.

[Kleinberg and Tardos, 2005] Kleinberg, J. and Tardos, E. (2005). *Algorithm Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[L. Palopol and Ursino, 1998] L. Palopol, D. S. and Ursino, D. (1998). Semi-automatic, semantic discovery of properties from database schemes. In *Proceedings of the 1998 International Symposium on Database Engineering & Applications*, IDEAS '98, pages 244–, Washington, DC, USA. IEEE Computer Society.

[Li et al., 2005] Li, Y., Liu, D.-B., and Zhang, W.-M. (2005). Schema matching using neural network. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Web Intelligence*, WI '05, pages 743–746, Washington, DC, USA. IEEE Computer Society.

[Madhavan et al., 2001] Madhavan, J., Bernstein, P. A., and Rahm, E. (2001). Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 49–58, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Manlove et al., 2002] Manlove, D., Irving, R. W., Iwama, K., Miyazaki, S., and Morita, Y. (2002). Hard variants of stable marriage. *Theor. Comput. Sci.*, 276(1-2):261–279.

[Melnik et al., 2002] Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Proceedings of the 18th International Conference on Data Engineering*, ICDE '02, page 117, Washington, DC, USA. IEEE Computer Society.

[Mesquita et al., 2007] Mesquita, F., Barbosa, D., Cortez, E., and da Silva, A. S. (2007). Fledex: flexible data exchange. In *Proceedings of the 9th annual ACM international workshop on Web information and data management*, WIDM '07, pages 25–32, New York, NY, USA. ACM.

[Popa et al., 2002] Popa, L., Hernadez, M. A., Velegrakis, Y., Miller, R., Naumann, F., and Ho, H. (2002). Mapping xml and relational schemas with clio. ICDE '02, pages 0–498, Los Alamitos, CA, USA. IEEE Computer Society.

[Pottinger and Bernstein, 2003] Pottinger, R. A. and Bernstein, P. A. (2003). Merging models based on given correspondences. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '2003, pages 862–873. VLDB Endowment.

[Quinlan, 1993] Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[Rahm and Bernstein, 2001] Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

[Seung et al., 1992] Seung, H. S., Opper, M., and Sompolinsky, H. (1992). Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, COLT '92, pages 287–294, New York, NY, USA. ACM.

[Shiang et al., 2008] Shiang, W.-J., Chen, H.-C., and Rau, H. (2008). An intelligent matcher for schema mapping problem. In *Proceedings of the Seventh International Conference on Machine Learning and Cybernetics*, volume 6 of *ICMLC'08*, pages 3172–3177.

[Wagner et al., 2011] Wagner, F., Macedo, J. A. F., and Lóscio, B. (2011). An incremental and user feedback-based ontology matching approach. In *Proceedings of the 13th International Conference on Information Integration and Web-based Applications and Services*, iiWAS '11, pages 371–374, New York, NY, USA. ACM.

[Witten et al., 2011] Witten, I. H., Frank, E., and Hall, M. A. (2011). *Data Mining: Practical Machine Learning Tools and Techniques - Third Edition*. Elsevier, San Francisco, CA, USA.

[Yang et al., 2008] Yang, Y., Chen, M., and Gao, B. (2008). An effective content-based schema matching algorithm. In *Proceedings of the 2008 International Seminar on Future Information Technology and Management Engineering*, FITME '08, pages 7–11, Washington, DC, USA. IEEE Computer Society.