UNIVERSIDADE DO VALE DO RIO DOS SINOS — UNISINOS
UNIDADE ACADÊMICA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO APLICADA
NÍVEL MESTRADO

TIAGO WIEDEMANN

SIMCOP: UM *FRAMEWORK* PARA ANÁLISE DE SIMILARIDADE EM SEQUÊNCIAS DE CONTEXTOS

SÃO LEOPOLDO
2014

Tiago Wiedemann

SIMCOP: UM *FRAMEWORK* PARA ANÁLISE DE SIMILARIDADE EM SEQUÊNCIAS DE CONTEXTOS

Dissertação apresentada como requisito parcial para a obtenção do título de Mestre pelo Programa de Pós-Graduação em Computação Aplicada da Universidade do Vale do Rio dos Sinos — UNISINOS

Orientador:
Prof. Dr. Jorge Luis Victória Barbosa

Co-orientador:
Prof. Dr. Sandro José Rigo

São Leopoldo
2014

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Wiedemann, Tiago

SIMCOP: Um *Framework* para Análise de Similaridade em Sequências de Contextos / Tiago Wiedemann — 2014.

125 f.: il.; 30 cm.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos, Programa de Pós-Graduação em Computação Aplicada, São Leopoldo, 2014.

"Orientador: Prof. Dr. Jorge Luis Victória Barbosa, Unidade Acadêmica de Pesquisa e Pós-Graduação".

1. Análise de Similaridade. 2. Sequências de Contextos. 3. Similaridade entre Sequências. 4. Similaridade Semântica. 5. Mineração de dados. I. Título.

CDU 004.732

Bibliotecária responsável: Carla Maria Goulart de Moraes — CRB 10/1252

Two roads diverged in a wood, and I.
I took the one less traveled by,
And that has made all the difference.
— ROBERT FROST

AGRADECIMENTOS

Dedico esta obra à minha esposa Deici, pelo companheirismo e pela paciência nestes anos de estudo, especialmente nesta etapa que se encerra agora. Agradeço por ter me incentivado a voltar a estudar após ter me afastado por alguns anos do meio acadêmico.

Agradeço também aos meus orientadores, Prof. Dr. Jorge L. V. Barbosa e Prof. Dr. Sandro J. Rigo, por toda a ajuda dada na escrita desta dissertação, desde a elaboração da proposta até o os últimos detalhes de fechamento do texto.

Por fim, gostaria de deixar meus agradecimentos também ao CNPq e a CAPES pelo apoio financeiro dado a esta pesquisa.

RESUMO

A Computação Ubíqua, que estuda formas de integrar a tecnologia ao cotidiano das pessoas, é uma área que vem crescendo nos últimos anos, especialmente devido ao desenvolvimento de tecnologias como a computação móvel. Um dos aspectos fundamentais para o desenvolvimento deste tipo de aplicação é a questão da Sensibilidade ao Contexto, que permite a uma aplicação adaptar o seu funcionamento conforme a situação na qual o usuário se encontra no momento. Com esta finalidade, diversos autores apresentaram definições formais sobre o que é um contexto e como representá-lo. A partir desta formalização começaram a ser desenvolvidas técnicas para análise de dados contextuais que propunham a realização de predições e inferências, entre outras análises. Esta dissertação especifica um *framework* denominado SIMCOP (*SIMilar Context Path*) para a realização da análise de similaridade entre sequências de contextos visitados por uma entidade. Este tipo de análise permite a identificação de contextos semelhantes com a intenção de prover funcionalidades como a recomendação de entidades e/ou contextos, a classificação de entidades e a predição de contextos. Um protótipo do *framework* foi implementado, e a partir dele foram desenvolvidas duas aplicações de recomendação, uma delas por um desenvolvedor independente, através do qual foi possível avaliar a eficácia do *framework*. Com o desenvolvimento desta pesquisa comprovou-se, conforme demonstrado nas avaliações realizadas, que a análise de similaridade de contextos pode ser útil em outras áreas além da computação ubíqua, como a mineração de dados e os sistemas de filtragem colaborativa, entre outras áreas, onde qualquer conjunto de dados que puder ser descrito na forma de um contexto, poderá ser analisado através das técnicas de análise de similaridade implementadas pelo *framework*.

Palavras-chave: Análise de Similaridade. Sequências de Contextos. Similaridade entre Sequências. Similaridade Semântica. Mineração de dados.

ABSTRACT

The Ubiquitous Computing, that studies the ways to integrate technology into the people's everyday life, is an area that has been growing in recent years, especially due to the development of technologies such as mobile computing. A key for the development of this type of application is the issue of context awareness, which enables an application to self adapt to the situation in which the user is currently on. To make this possible, it was necessary to formally define what is a context and how to represent it . From this formalization, techniques for analyzing contextual data have been proposed for development of functions as predictions or inferences. This paper specifies a framework called SIMCOP (SIMilar Context Path) for performing the analysis of similarity between sequences of contexts visited by an entity. This type of analysis enables the identification of similar contexts with the intention to provide features such as the recommendation of entities and contexts, the entities classification and the prediction of contexts. The development of this research shows that the contexts similarity analysis can be useful in other areas further the ubiquitous computing, such as data mining and collaborative filtering systems. Any data type that can be described as a context, can be analyzed through the techniques of similarity analysis implemented by the framework, as demonstrated in the assessments.

Keywords: Similarity Analysis. Context Sequences. Sequences Similarity. Semantic Similarity. Data Mining.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

Config.	Configuração
Desenv.	Desenvolvimento
Seq.	Sequência
Traj.	Trajetória ou *Trajectory*

LISTA DE SIGLAS

ACM	Association for Computing Machinery
BOC	Bag-Of-Concepts
BOW	Bag Of Words
CEP	Complex Event Processing
DTW	Dynamic Time Warping
ED	Entity Description
GUID	Globally Unique IDentifier
HTML	HyperText Markup Language
IEEE	Institute of Electrical and Electronics Engineers
KDD	Knowledge Discovery in Databases
LBSN	Location Based Social Network
LCSS	Longest Common Subsequence
OASIS	Organization for the Advancement of Structured Information Standards
OOP	Object-Oriented Programming
OWL	Web Ontology Language
PCA	Principal Component Analisys
PDA	Personal Digital Assistant
PDF	Probability Density Function
PIPCA	Programa Interdisciplinar de Pós-graduação em Computação Aplicada
POI	[Places / Points] Of Interest
RDF	Resource Description *Framework*
RIF	Rule Interchange Format
RF	Requisitos Funcionais
RNF	Requisitos Não Funcionais
RV	Realidade Virtual
SGBD	Sistema Gerenciador de Banco de Dados
TOI	Time Of Interest
UML	Unified Modeling Language
UOI	URL's Of Interest
URI	Universal Resource Identifier
URL	Uniform Resource Locators
VANET	Vehicular Ad-hoc NETwork
XML	eXtensible Markup Language
W3C	World Wide Web Consortium

SUMÁRIO

1 INTRODUÇÃO

A área da Computação Ubíqua, segundo WEISER (1991), estuda o desenvolvimento de técnicas que visam integrar perfeitamente a tecnologia da informação ao cotidiano das pessoas. Isto não deve ser confundido com o conceito de Realidade Virtual (RV), que segundo o autor é uma ideia totalmente oposta ao que ele propõe. Enquanto a RV tenta trazer o mundo real para dentro de uma simulação, a proposta de Mark Weiser é justamente que as pessoas sejam auxiliadas pela tecnologia no mundo real, de forma pró-ativa, enquanto realizam suas atividades.

Um aspecto vital para que esta visão se torne realidade é a possibilidade de desenvolver aplicações sensíveis ao contexto. Segundo DEY; ABOWD; SALBER (2001) entende-se por contexto qualquer informação que permita caracterizar a situação de entidades que sejam relevantes para a interação entre um usuário e uma aplicação. Isto significa em geral informações sobre a situação, identidade e localização espaço-temporal de pessoas, grupos e objetos físicos ou computacionais. Através do conhecimento de dados contextuais, uma aplicação pode ajustar seu próprio funcionamento ou ainda agir de maneira pró-ativa, alertando o usuário de algum perigo ou ajudando-o a realizar suas atividades de forma mais eficiente.

A capacidade de reconhecer a situação atual de um usuário e o estado dos objetos e pessoas próximas abre algumas possibilidades de pesquisas. Uma questão relevante é sobre como analisar as sequências de contextos capturados e armazenados por diferentes tipos de aplicações. Esta dissertação aborda um tipo específico de análise de dados em sequências de contextos, que é a análise da similaridade entre sequências de contextos de entidades genéricas. Com este propósito, é proposto um *framework* denominado *SIMCOP* (*SIMilar COntext Path framework*) que permite combinar diferentes técnicas de similaridade para analisar os diversos tipos de contextos visitados por uma entidade.

1.1 Motivação

O desenvolvimento de aplicações sensíveis ao contexto é um dos principais desafios na área de Computação Ubíqua. Segundo SATYANARAYANAN (2001) um sistema de Computação Ubíqua que pretenda atender ao requisito de agir de forma pró-ativa mas não-intrusiva, deve possuir recursos para reconhecimento do contexto atual. Esta capacidade permitiria ao sistema identificar, por exemplo, que uma informação relevante para a atividade atual do usuário está disponível, mas que no momento a situação do usuário não permite que ele seja interrompido, portanto o sistema poderá tomar a decisão de aguardar um momento mais oportuno para notificá-lo.

Com o desenvolvimento deste tipo de aplicação começou a surgir a necessidade de representar formalmente os contextos e armazená-los, gerando assim diversas bases de dados que armazenam as sequências de contextos capturados ao longo do tempo. Pesquisadores vêm estudando as oportunidades que podem surgir a partir da análise destas bases. ROSA (2013) por

exemplo, desenvolveu um modelo chamado ORACON que prevê os contextos futuros de uma entidade a partir de inferências sobre seu histórico de contextos. Já SILVA et al. (2010) chamam estas sequências de trilhas, e propõem um modelo chamado UbiTrail, acessado por aplicações clientes que necessitem obter e analisar dados sobre as trilhas.

Uma maneira de se analisar as sequências de contextos é tratá-las como um tipo de série temporal. Segundo SOUZA (1989), séries temporais são sequências de dados numéricos, ordenados no tempo e obtidos através de observações. Se os dados disponíveis nos contextos analisados forem constituídos de valores numéricos ou forem quantificáveis, será possível aplicar as técnicas de análise de séries temporais às sequências de contextos. Os dados contextuais serão sempre oriundos de observações da situação de uma entidade, coletados via sensores físicos, registros de *software*, ou qualquer outro mecanismo. E estarão ordenados no tempo, pois mesmo que a data e a hora não estejam registradas, a ordem de coleta das informações contextuais é preservada, o que garante que a ordenação cronológica da sequência de contextos.

Uma informação útil a ser extraída da análise de sequências de contextos é a similaridade entre duas sequências. Esta informação poderia, por exemplo, ajudar a encontrar grupos de entidades que possuem sequências de atividades semelhantes, ou encontrar padrões de semelhança entre entidades que frequentam um determinado local. Esta porém, não é uma tarefa trivial, pois a natureza dos dados contextuais tende a ser heterogênea, combinando dados de diversas fontes, com formatos e significados diferentes.

Cada categoria de dados disponível no contexto pode exigir métricas de similaridade específicas. Uma comparação precisa e útil da posição de duas pessoas, por exemplo, exige que sejam consideradas as particularidades dos dados geográficos. Por outro lado a comparação dos sinais vitais destas mesmas pessoas necessita ser feita considerando o conhecimento médico existente. Esta característica dos contextos exige que técnicas diferentes sejam combinadas para se chegar a um valor de similaridade global dos contextos analisados.

Outro aspecto importante para a similaridade é a questão da interpretação semântica dos dados obtidos a partir dos contextos visitados. A leitura puramente léxica destes dados pode ocultar informações importantes, que permitiriam uma análise mais precisa da similaridade das sequências. Por exemplo, suponha que dois usuários cheguem aos seus escritórios, aproximadamente no mesmo horário e liguem seus computadores. Um deles abre um arquivo no *MS-Word*™e o outro abre um arquivo no *Open Office Writer*. Um *software* que analisasse de forma automática os *logs* destes computadores não apontaria nenhuma semelhança entre estes dois usuários, visto que os nomes das aplicações são diferentes. Técnicas baseadas em encontrar semelhanças léxicas em palavras também não obteriam sucesso, já que os nomes destas aplicações não apresentam semelhança gramatical. No entanto, se houver alguma forma de se mapear os relacionamentos semânticos entre estes termos, seria possível ao *software* deste exemplo encontrar um nível de relação entre ambas as aplicações. Identificando por exemplo, que tratam-se de *softwares* para edição de textos.

Este mapeamento semântico pode ser conseguido através das Ontologias. Segundo GRU-

BER (1995), uma ontologia é uma especificação explícita de uma conceitualização. O termo foi tomado emprestado da filosofia porém com um significado diferente. Em sistemas de representação do conhecimento, uma ontologia pode ser descrita como a especificação formal de conceitos sobre um determinado domínio, e os relacionamentos entre estes conceitos. No exemplo anterior os conceitos *MS-Word*™e *Open Office Writer* teriam ambos um relacionamento do tipo 'é um' com o conceito 'Editores de Texto', o que permitiria descobrir que há um determinado grau de similaridade entre ambos.

1.2 Definição do Problema

Os fatores apontados na seção 1.1 tornam a análise de similaridade em sequências de contextos uma questão desafiadora para a pesquisa tanto na área de Computação Ubíqua quanto na engenharia de *software* em geral. Diversos problemas devem ser tratados em *softwares* que necessitem deste tipo de análise, como por exemplo:

- É possível quantificar a similaridade?
- Qual a melhor métrica de similaridade para cada caso?
- Existe uma métrica única de similaridade que trate todos os casos?
- Como lidar com a natureza heterogênea dos dados contextuais?
- Como tratar as variações dos dados contextuais ao longo do tempo, para se chegar a um valor de similaridade global entre duas sequências?
- Como utilizar as ontologias para determinação de similaridades semânticas?
- Como identificar quais informações são mais relevantes para a análise de similaridade?

Estas dificuldades justificam a especificação e implementação de um *framework* adaptável capaz de encapsular diferentes técnicas de análise de similaridade e oferecer uma interface unificada para aplicações.

1.3 Objetivos

Este trabalho de pesquisa tem como objetivo geral a especificação de um *framework* extensível e configurável, que permita combinar diferentes técnicas para análise de similaridade em sequências de contextos. Para se atingir este objetivo geral, são definidos os seguintes objetivos específicos:

- Avaliar os fundamentos teóricos da área;
- Identificar e comparar os trabalhos relacionados;
- Definir a especificação do *framework*;

- Implementar um protótipo do *framework*;
- Validar a especificação a partir do protótipo.

1.4 Metodologia

Para a realização desta pesquisa, inicialmente procurou-se identificar quais seriam as tecnologias e tópicos que oferecessem o embasamento teórico necessário para a concepção de de *framework* que atendesse aos objetivos propostos. Foram realizadas pesquisas nas áreas que possuem relação com o tema, ou poderiam ajudar de alguma forma na resolução do problema identificado. Este embasamento prévio foi importante para a construção de uma proposta que trouxesse embutida o conhecimento prévio obtido pelas linhas de pesquisa consultadas.

Em seguida, partiu-se para a escrita de uma especificação inicial, para que fosse possível especificar com o máximo de detalhamento possível os requisitos funcionais e não-funcionais do *framework* a ser desenvolvido. Também foi desenvolvida uma proposta inicial de arquitetura que atendesse as demandas do problema e atingisse os objetivos e requisitos identificados.

Partindo deste embasamento teórico e do levantamento inicial de requisitos, buscou-se identificar trabalhos semelhantes, para estudar as abordagens adotadas e os resultados obtidos. Foi então realizado um comparativo entre estes trabalhos de forma a identificar os pontos fortes e fracos de cada abordagem e com isso aperfeiçoar a especificação do *framework*.

Obtido este panorama detalhado elaborou-se a especificação do *framework*, a partir do qual foi implementado um protótipo com o qual foram desenvolvidos dois sistemas de recomendação, que serviram para avaliar sua eficácia considerando-se as opiniões dos usuários em relação a relevância das recomendações efetuadas e a corretude das semelhanças encontradas pelas funções de análise de similaridade implementadas.

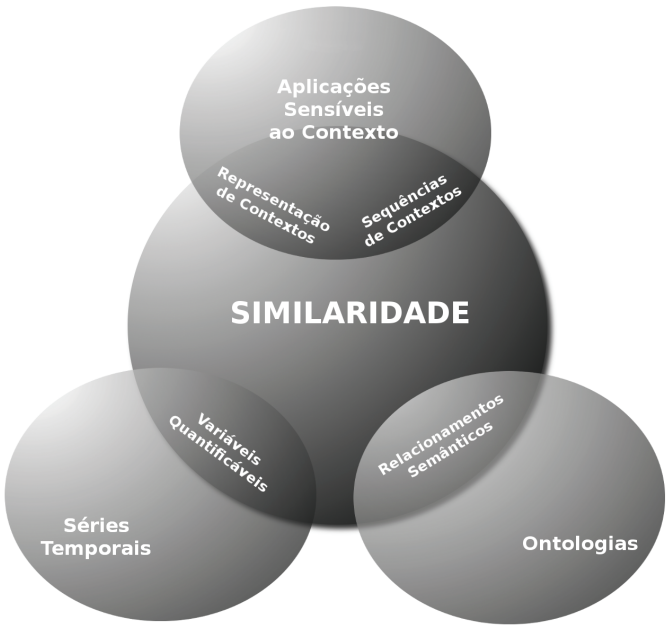
1.5 Organização do Texto

Esta dissertação está dividida em seis capítulos. No segundo capítulo serão discutidos os tópicos de pesquisa que são relevantes para o trabalho proposto. No terceiro capítulo são apresentados os trabalhos relacionados e realizado um estudo comparativo de suas características, funcionalidades e limitações. A especificação do *framework*, com sua arquitetura e funcionalidades será apresentado no quarto capítulo. O quinto capítulo discute a implementação do protótipo e o resultado das avaliações realizadas. Por fim, o sexto e último capítulo apresenta as considerações finais e aponta os trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentado um resumo das referências teóricas consultadas para a elaboração desta pesquisa. A Figura 1 ilustra como os conceitos estudados se relacionam para compor o embasamento que levou a especificação do *framework* proposto. A área de aplicações sensíveis ao contexto, que é relevante para a Computação Ubíqua, oferece os meios para representar os dados contextuais e gerenciar as sequências de contextos visitadas por uma entidade (SATYANARAYANAN, 2001; DEY; ABOWD; SALBER, 2001; WEISER, 1991). Diferentes técnicas são utilizadas para analisar a similaridade, conforme o tipo de informação presente no contexto. Quando o contexto estudado é composto exclusivamente por dados numéricos, ou se a análise que se pretende realizar permite que os contextos observados sejam sumarizados por estatísticas, é possível aplicar as técnicas de análise de similaridade em séries temporais (SOUZA, 1989). Por outro lado, caso deseje-se analisar os dados contextuais levando-se em conta o seu significado, é possível a utilização de ontologias para mapear os relacionamentos semânticos entre termos presentes na descrição dos contextos (GRUBER, 1995; WANG et al., 2004).

Figura 1: Temas relacionados



Fonte: Elaborado pelo autor

O texto deste capítulo está organizado em 8 seções. A primeira seção é um breve histórico da área da Computação Ubíqua e como ela depende da sensibilidade ao contexto. A segunda trata das aplicações sensíveis ao contexto. A terceira seção discute os conceitos de trilhas, histórico de contextos e sequência de contextos, suas semelhanças, diferenças e a justificativa da escolha do termo 'sequência de contextos' para este trabalho. Na quarta seção são estudadas as formas de representação de dados contextuais. A quinta seção trata do tema Ontologias e analisa as técnicas para se mapear os relacionamentos semânticos entre os termos presentes na descrição

dos contextos. A sexta seção apresenta uma introdução ao estudo de séries temporais. Na sétima seção é apresentado o conceito formal de similaridade adotado para esta dissertação, e realizado um resumo das principais métricas de similaridade aplicáveis às sequências de contextos. Por fim, a oitava seção resume os temas abordados.

2.1 Computação Ubíqua

O conceito de Computação Ubíqua foi apresentado pela primeira vez por WEISER (1991), pesquisador do Laboratório de Ciência Computacional da *Xerox Palo Alto Research Center*, em seu clássico artigo *'The Computer for the 21th century'*. Para o autor, as tecnologias que causam mais impacto na vida das pessoas são aquelas que de certa forma desaparecem, ou seja, estão tão integradas ao cotidiano que são utilizadas de forma inconsciente.

O autor cita o exemplo da escrita, uma tecnologia que permite transmitir ideias de geração em geração e que supera a capacidade de memória humana. Esta é uma tecnologia ubíqua, não apenas por estar disponível a todo momento e em qualquer lugar, mas principalmente por que ao fazer uso dela toda a atenção dos leitores está focada no conteúdo do texto e não no processo de tradução de símbolos em ideias. Neste sentido pode-se dizer que a escrita é uma tecnologia invisível, já que não é necessário prestar atenção na operação da mesma para poder usufruir de seus benefícios.

Segundo SATYANARAYANAN (2001) a área da Computação Ubíqua é um evolução de uma linha de pesquisa que iniciou-se ainda na década de 1970, com a Computação Distribuída. Inicialmente, preocupava-se com questões como processamento remoto, tolerância a falhas, alta disponibilidade e segurança. A partir da década de 1990 com o advento da computação móvel, novos desafios surgiram, como o consumo de bateria e pouca robustez dos equipamentos. Além disso, a comunicação sem fio traz a questão da variabilidade na qualidade do sinal, o que exige protocolos de tratamento de erros mais robustos entre outros desafios. A evolução da tecnologia de computação móvel começa a tornar possível que a visão de WEISER (1991) sobre uma tecnologia invisível, totalmente integrada ao cotidiano e disponível a todo momento em qualquer lugar, se torne realidade.

Um ponto destacado tanto por WEISER (1991) quanto por SATYANARAYANAN (2001) é que a Computação Ubíqua não trata somente de mobilidade, mas principalmente da construção de sistemas computacionais que conseguem se adaptar ao estilo de vida de seus usuários, inclusive podendo tomar decisões de forma pró-ativa para auxiliar o usuário nas atividades em que ele está envolvido.

Um conceito chave é a questão da sensibilidade ao contexto. A possibilidade de uma aplicação conseguir coletar dados sobre a situação atual do ambiente na qual opera, e a partir do conhecimento do perfil do usuário, ser capaz de adaptar seu funcionamento está no cerne do conceito de Computação Ubíqua.

2.2 Aplicações Sensíveis ao Contexto

Para BROWN; BOVEY; CHEN (1997) aplicações sensíveis ao contexto são aquelas cujo funcionamento é orientado principalmente pelo contexto atual do usuário. Estas podem ser divididas basicamente em *contínuas* e *discretas*. Nas aplicações contínuas as informações apresentadas ao usuário são atualizadas constantemente. Os autores citam como exemplo, um sistema de localização que continuamente exibe ao usuário uma seta indicando a direção que ele deve tomar para chegar a um determinado destino. A direção apontada pela seta varia conforme o usuário se movimenta. Em aplicações discretas, conjuntos independentes de informações estão associados a diferentes contextos, e são exibidos sempre que o usuário entra em um novo contexto. Segundo os autores a maioria das aplicações se enquadram melhor no conceito discreto.

A definição do conceito de contexto varia conforme com o autor. Para SCHILIT; THEIMER (1994), por exemplo, o contexto é constituído por dados sobre a localização, identidade de pessoas e objetos próximos e as alterações nestes objetos. Em BROWN; BOVEY; CHEN (1997) são acrescentados ainda dados ambientais como a hora do dia, estação do ano e temperatura. O estudo de PASCOE; RYAN; MORSE (1998) sobre quais seriam as necessidades de pesquisadores de campo, como arqueólogos e biólogos, na utilização de PDA's considera que o contexto deve ter informações sobre identidade, localização, ambiente e tempo, já que para estes autores as atividades do usuário estão intimamente ligadas ao contexto no qual ele se encontra. Na definição de DEY; ABOWD; WOOD (1998), o contexto é qualquer informação que possa ser usada para melhorar a experiência do usuário, o que implica em informações como o estado emocional do usuário, sua localização física, o ambiente social, objetos e pessoas próximas, entre outras.

Segundo DEY; ABOWD; SALBER (2001), estes trabalhos definem o conceito de contexto através de exemplos de aplicações. O que os torna específicos para determinados domínios, dificultando sua aplicação em outras áreas já que não deixam claro como saber se um novo tipo de informação pode ser classificado como contexto. Por outro lado, segundo os autores, definições mais genéricas como as propostas por BROWN (1996), FRANKLIN; FLASCHBART (1998), WARD; JONES; HOPPER (1997) e RODDEN et al. (1998) acabam sendo difíceis de serem implementadas pois não determinam como identificar os elementos que constituem um contexto.

Para SCHILIT; ADAMS; WANT (1994), os aspectos mais importantes de um contexto são: aonde o usuário está, com quem o usuário está e quais recursos estão próximos. Os autores ainda dividem o ambiente de execução em Ambiente Computacional, como processadores disponíveis, dispositivos acessíveis, rede, conectividade e custos computacionais; Ambiente do Usuário, como localização, pessoas próximas e situação social; Ambiente Físico, como luminosidade e nível de ruído.

Para esta dissertação será adotada a seguinte definição de contexto:

"Contexto é qualquer informação que possa ser usada para caracterizar a situação de entidades que são consideradas relevantes para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação. Contextos são, tipicamente: a localização, identidade e o estado de pessoas, grupos e objetos físicos e computacionais" (DEY; ABOWD; SALBER, 2001).

Para estes autores, o objetivo de conhecer o contexto é determinar qual atividade o usuário está tentando realizar. Já que é difícil determinar diretamente quais são os objetivos do usuário, dados do contexto podem ajudar a inferir esta informação.

Outra questão levantada por DEY; ABOWD; SALBER (2001), é a da diversidade de informações que podem compor o contexto. Para lidar com esta diversidade, é recomendável que se busque uma forma de categorização dos dados contextuais, para facilitar sua compreensão. Os autores propõem uma classificação simples destas informações, baseada nas entidades envolvidas no contexto avaliado. As entidades mais importantes identificadas pelos autores são: lugares, pessoas e coisas. **Lugares** são regiões de espaço geográfico como salas e construções. **Pessoas** podem ser tanto indivíduos como grupos compartilhando a mesma região ou distribuídos. **Coisas** podem tanto ser objetos físicos como componentes de *software* ou artefatos.

Também são definidas quatro categorias essenciais da informação de contexto: Identidade, localização, estado ou atividade e tempo. **Identidade** refere-se a capacidade de atribuir um identificador único a cada entidade. **Localização** é mais do que simplesmente a posição atual da entidade. Pode abranger também a dados como a orientação e elevação, bem como informações que permitam deduzir as relações espaciais entre as entidades. Por exemplo, a possibilidade de determinar se um objeto está acima ou abaixo de outro. O conceito também se aplica a lugares e podem ter coordenadas relativas e absolutas. **Situação** ou **atividade** refere-se a características inerentes à entidade vinculada ao contexto. Para lugares, podem ser informações como a temperatura, umidade, luminosidade ou nível de ruído. Para pessoas ou grupos pode significar o estado emocional, seus sinais vitais ou ainda a atividade na qual o usuário está envolvido. Para coisas, podem ser dados de sensores ou no caso de *softwares*, podem ser atributos como o uso de CPU ou de memória. Por fim, o **tempo** identifica o momento em que os eventos ocorreram ou que os dados contextuais tenham sido capturados, permitindo a análise histórica destes eventos. Assim como a localização, o tempo possui um significado mais amplo do que aparenta inicialmente. Podendo representar um instante específico ou um intervalo, mas também ser simplesmente um valor sequencial indicando a ordem em que os eventos ou as coletas de dados do contexto ocorreram.

Como justificativa para a escolha destas categorias os autores argumentam que através dos dados fornecidos por elas é possível inferir informações adicionais, estendendo o conhecimento sobre a situação. Por exemplo, o endereço de um usuário poderia ser derivado a partir da identidade, ou ainda, a quantidade de pessoas em uma sala em determinado momento permitiria deduzir se uma reunião irá ou não ocorrer.

A capacidade de organizar e armazenar os dados contextuais vinculados a uma entidade, cria o conceito de histórico de contextos, que é discutido na próxima seção.

2.3 Trilhas, Históricos ou Sequências de Contextos

Talvez o primeiro trabalho em ciência da computação a apresentar o conceito de trilhas como uma sequência de itens visitados por um usuário, seja o de BUSH; WANG (1945). Os autores previram que os computadores seriam uma ferramenta poderosa para organizar o conhecimento humano. Em seu trabalho é proposta uma máquina, denominada *Memex*, que armazenaria uma grande quantidade de documentos no formato de microfilmes. O usuário selecionaria e consultaria estes documentos em um sistema que permitiria inserir anotações criando vínculos (*links*) arbitrários, gerando assim uma trilha de documentos semanticamente encadeados conforme a preferência do usuário. Esta provavelmente é a primeira descrição do hipertexto, um conceito que só se popularizou no final do século XX.

Para DRIVER; CLARKE (2004) uma trilha é, no nível mais geral, uma coleção de localizações acompanhadas de informações associadas e uma ordem recomendada de visitação. Os autores utilizam a metáfora da trilha para capturar as atividades diárias de um usuário e adaptar o funcionamento de aplicações móveis sensíveis ao contexto, considerando-se o histórico de atividades e localizações visitados pelo usuário. Aplicações sensíveis as trilhas devem processar um conjunto de informações contextuais, como condições do trânsito em diferentes rodovias, meios de transporte e condições do tempo, para oferecer serviços como, por exemplo, a sugestão de rotas para motoristas em cidades desconhecidas ou gerenciar a logística de serviços de entrega.

SPENCE; DRIVER; CLARKE (2005) estendem o trabalho de (DRIVER; CLARKE, 2004), ao considerar que é possível analisar também o histórico de trilhas recomendadas. Os autores chamam de *trail histories* as informações sobre como o usuário fez uso das trilhas sugeridas no passado. Isto inclui informações como a trilha inicial proposta pela aplicação, todas as reorganizações de trilhas que ocorreram, a trilha atual seguida pelo usuário, e outras informações do contexto, que podem ajudar a identificar por que o usuário não seguiu a trilha original.

Posteriormente, em DRIVER; CLARKE (2008) uma trilha é descrita como uma coleção de atividades a serem executadas pelo usuário, semelhante a uma *'to-do list'* mas cuja sequência de execução é ordenada conforme o contexto e as preferências do usuário. Os autores argumentam que a trilha pode ser usada como um modelo genérico capaz de satisfazer os requisitos de aplicações de gerenciamento de tempo sensíveis ao contexto. As trilhas superam as limitações das técnicas tradicionais deste tipo de aplicação, geralmente baseada em listas estáticas de atividades, substituindo-as por listas reordenadas dinamicamente de acordo com o contexto do usuário. Partindo deste princípio, os autores apresentam um *framework* que visa oferecer funcionalidades para a geração de trilhas em resposta a alterações no contexto atual do usuário.

A análise destes dados pode auxiliar no desenvolvimento de aplicações de contextos colaborativos. Para os autores, o conceito de *collaborative context* refere-se ao compartilhamento de dados entre dispositivos em ambientes de Computação Ubíqua. Através da análise dos históricos de trilhas, algumas funcionalidades deste tipo de aplicação podem ser aperfeiçoadas, como

por exemplo, impedir gerações desnecessárias de trilhas, a verificação de hipóteses e adaptação baseada no *feedback* do usuário.

Em sua tese de doutorado, SMITH (2008) sugere um método para registrar anotações semânticas sobre a vida de usuários, utilizando uma combinação de dados pessoais coletados em sistemas ubíquos e dados públicos disponíveis livremente na *web*. O autor utiliza o mesmo conceito de anotação utilizado na ideia original da Web Semântica (W3C - World Wide Web Consortium, 2013a), onde páginas *web* recebem anotações que permitem descrever formalmente os dados apresentados. O autor argumenta que as técnicas de computação sensível ao contexto possuem o potencial de gerar anotações sobre a vida de uma pessoa, criando assim um histórico de locais e situações vividos pelo usuário.

Para SILVA et al. (2010) o trabalho de SMITH (2008) demonstra que o registro das atividades executadas por um usuário, acompanhado de informações sobre a localização, permite gerar históricos de contextos visitados ao longo do tempo. Este histórico também pode ser chamado de trilha, como em DRIVER; CLARKE (2008). SILVA et al. (2010) propõem um modelo de gerenciamento de trilhas através de uma arquitetura de serviços disponibilizados para aplicações clientes.

Recentemente, LI; EICKHOFF; VRIES (2012) denominam como trilha a sequência de locais de interesse (*POI - Places of Interest*) que um usuário visita durante um dia. Os autores propõem um modelo para prever quais serão os tipos de locais que farão parte da trilha do usuário no futuro, baseado no histórico de locais visitados armazenado na trilha.

Percebe-se que estes trabalhos têm em comum o fato de lidarem com registros de sequências de eventos, ordenados cronologicamente e vinculadas a uma entidade identificável. A diferença está no tipo de informação que é descrita nestas sequências. Alguns destes trabalhos lidam sequências que descrevem a Localização (DRIVER; CLARKE, 2004; LI; EICKHOFF; VRIES, 2012) ou Atividade de um Usuário (DRIVER; CLARKE, 2004, 2008; SMITH, 2008), enquanto que (SILVA et al., 2010) busca oferecer suporte a entidades genéricas. Isto corresponde as categorias propostas por DEY; ABOWD; SALBER (2001) para descrever o contexto de uma entidade. O fato das trilhas destes trabalhos estarem em ordem cronológica e estarem vinculadas a uma entidade ou usuário que pode ser identificado, corresponde respectivamente as categorias Tempo e Identidade. A atividade de um usuário é uma das informações possíveis de serem descritas na categoria Estado/Situação. Por fim os dados que permitem identificar onde o usuário se encontra se enquadram na categoria Localização. Portanto, conclui-se que a definição de trilha adotada por estes autores pode ser generalizada como Histórico ou Sequência de Contextos.

Com base no exposto, será adotada nesta dissertação o termo ‘sequência de contextos’ para se referir a lista cronológica de contextos que foram, ou serão visitados por uma entidade identificada. O termo ‘trilha’, como visto no início desta seção, pode referir-se a qualquer sequência de itens, como no caso dos documentos encadeados de BUSH; WANG (1945), e o foco desta dissertação é especificamente em trilhas de contextos. Por outro lado, o termo ‘histórico de

contextos' implica em análise de eventos ocorridos no passado, o que nem sempre corresponde ao problema que deverá ser tratado pelo modelo. Por exemplo, o *framework* desenvolvido por DRIVER; CLARKE (2008) produz sugestões de listas de atividades a serem executadas. Caracterizando-se portanto como uma 'trilha de contextos futuros' de um usuário.

2.4 Ontologias

O dicionário Aurélio define ontologia como sendo a 'parte da filosofia que trata da natureza do ser' (HOLANDA FERREIRA, 2008). O termo em latim 'ontologia' como 'ciência do ser' foi inventado pelo filósofo alemão Jacob Lorhard, em seu livro *Ogdoas Scholastica* em 1606. O filósofo Christian Wolff argumenta que a ontologia é uma disciplina que revela a essência das coisas, uma visão criticada por David Hume e Immanuel Kant. No começo do século XX, o termo foi adaptado pelo fundador da Fenomenologia, Edmund Husserl, que denominou a metafísica de Wolff como 'Ontologia Formal', em contraste com as ontologias 'regionais', como as da matemática, mente, cultura e religião (ENCYCLOPEDIA BRITANNICA, 2013).

No contexto da ciência da computação, GRUBER (2008) define ontologia como um conjunto de primitivas representacionais com as quais é possível modelar o conhecimento ou um discurso. Em geral, segundo o autor, estas primitivas correspondem a classes, atributos e relacionamentos entre classes. Nesta definição estão incluídas informações sobre o significado e também as restrições quanto a implementação em aplicações logicamente consistentes. Ontologias são especificadas tipicamente em linguagens que permitem abstrair a estrutura dos dados e detalhes de sua implementação. Por esta razão é dito que ontologias encontram-se no nível semântico, em contraste com outras formas de modelagem de dados, como *Database Schemas*, que são especificados no nível lógico ou físico.

Esta capacidade de se descrever os relacionamentos semânticos entre conceitos de um domínio torna a utilização de ontologias uma ferramenta útil para a análise de similaridade em sequências de contextos, principalmente quando os dados contextuais forem compostos por descrições ou categorias e deseja-se analisar a similaridade do ponto de vista semântico.

2.4.1 Web Semântica

A utilização de ontologias vem se popularizando com o surgimento do conceito de Web Semântica (BERNERS-LEE; HENDLER; LASSILA, 2001). Segundo estes autores, a Web tradicional surgiu como um ambiente para armazenamento de hipertexto escrito em HTML, uma forma eficaz de descrever o conteúdo e a apresentação de um documento, porém ineficaz para descrever o significado dos dados apresentados. Para superar esta limitação foi proposta a Web Semântica, como uma extensão da Web tradicional, onde além dos dados serem descritos de forma legível a humanos, também o seriam para as máquinas.

Para alcançar este objetivo os computadores precisam operar sobre coleções de informações

processadas através de conjuntos de regras de inferência. O problema é que o conteúdo da Web é produzido de forma descentralizada, portanto cada usuário pode descrever seus dados de forma totalmente arbitrária, impossibilitando que haja qualquer tipo de padrão centralizado para troca de informações. Para BERNERS-LEE; HENDLER; LASSILA (2001) três tecnologias são fundamentais para contornar esta limitação e permitir o compartilhamento de informações em um ambiente heterogêneo: XML, RDF e Ontologias.

A linguagem XML permite que usuários descrevam informações na forma de texto estruturado em *tags*. Isto permite identificar as partes de uma informação, porém como o criador de um documento XML pode definir suas próprias *tags*, ainda não é possível identificar o significado dos dados. Para suprir esta necessidade, utiliza-se o *Resource Description Framework* (RDF): uma linguagem para formalização do significado da estrutura de um documento. Uma sentença RDF é formada por uma tupla composta por *subject*, *verb* e *object*, que é utilizada para montar afirmações sobre recursos (pessoas, ou artefatos) que possuem propriedades (como 'é irmão de', 'é autor de'), com determinados valores. Um ponto crucial para a interoperabilidade é que o *subject* e o *object* sejam codificados em uma URI, um formato de identificação de recursos cujo exemplo mais popular são as URL's utilizadas para identificar endereços de *sites*.

Mas, mesmo a utilização de URI's ainda não é suficiente para garantir a capacidade de processamento sobre a semântica de um documento, pois diferentes autores podem utilizar diferentes identificadores para o mesmo conceito. Por exemplo, a informação do CEP de um endereço seria descrita por um autor americano, provavelmente como *zip code*. Para permitir que máquinas sejam capazes de processar corretamente estas informações, é necessário que haja uma estrutura formal descrevendo os termos e suas relações entre si. Ainda segundo o exemplo anterior, tanto *zip code* quanto o CEP são um 'código de endereçamento postal' que, por sua vez, faz parte de um 'endereço'. É para este fim que são utilizadas as ontologias.

Uma ontologia pode ser entendida como uma taxonomia, ou seja uma estrutura de classes, enriquecida com outros tipos de relações além da relação 'é um' descrita por estruturas deste tipo. No exemplo acima, 'Código de Endereçamento Postal' não é uma classe filha de 'Endereço', porém está relacionada a esta através de uma relação do tipo 'faz parte de'. Esta classe ainda poderia ter uma propriedade do tipo 'é utilizada em', indicando o país onde é utilizada, e suas classes filhas *'zip code'* e 'CEP' teriam restrições como: 'é utilizada em → somente → *USA*', e 'é utilizada em → somente → Brasil', respectivamente.

A linguagem OWL (*Web Ontology Language*) é um padrão para descrição formal de ontologias, criado pelo W3C - World Wide Web Consortium (2013a) e baseado na linguagem DAML+OIL e na lógica de descrição *(W3C - World Wide Web Consortium, 2013b)*. Foi publicada inicialmente em 2004 e expande as funcionalidades do RDF, acrescentando a capacidade de expressar relacionamentos semânticos de maneira formal. A OWL possui três sublinguagens ou espécies:

1. **OWL Lite** voltada para aplicações simples que necessitam apenas de uma maneira de classificar dados de forma hierárquica e utilizem apenas mecanismos simples de restri-

ções;

2. **OWL DL** utilizada em cenários onde é necessário a máxima expressividade porém sem perder a capacidade de computabilidade da ontologia descrita;
3. **OWL Full** permite a máxima expressividade e liberdade de sintaxe, porém não garante a computabilidade.

2.4.2 Inferência em Ontologias

O conceito de inferência em ontologias para aplicações de Web Semântica pode ser definido, em linhas gerais, como a aplicação de processos automáticos capazes de gerar novos relacionamentos baseados nos dados e em algumas informações adicionais como por exemplo, um conjunto de regras de inferência. Enquanto as ontologias são normalmente desenvolvidas com o foco de classificação de dados, as regras de inferência focam-se em definir mecanismos gerais para descoberta e geração de novos relacionamentos baseados nos relacionamentos já existentes. Por exemplo, se um conjunto de dados define um relacionamento do tipo Flipper é um Golfinho e uma ontologia defina que todo Golfinho é um Mamífero, então um motor de regras de inferência que conheça a regra *'se A $\Rightarrow$ B e B $\Rightarrow$ C então A $\Rightarrow$ C'*, seria capaz de criar o relacionamento 'Flipper é um Mamífero', mesmo que este relacionamento não faça parte do conjunto de dados ou das ontologias originais (W3C - World Wide Web Consortium, 2013c).

Como a linguagem OWL foi projetada baseada na Lógica de Descrição (HORROCKS et al., 2007), é possível explorar as técnicas de inferência já desenvolvidas nesta área para a descoberta de conhecimento em ontologias. WANG et al. (2004) demonstra algumas regras de inferência suportadas pela linguagem OWL-Lite, conforme a Tabela 1.

Tabela 1: Lista parcial de regras de inferência em ontologias OWL

OWL	**Regra**
Transitive-Property	(?P rdf:type owl:TransitiveProperty) $\wedge$ (?B ?P ?C) $\wedge$ (?A ?P ?B) $\Rightarrow$ (?A ?P ?C)
subClassOf	(?a rdfs:subClassOf ?b) $\wedge$ (?b rdfs:subClassOf ?c) $\Rightarrow$ (?a rdfs:subClassOf ?c)
subPropertyOf	(?a rdfs:subPropertyOf ?b) $\wedge$ (?b rdfs:subPropertyOf ?c) $\Rightarrow$ (?a rdfs:subPropertyOf ?c)
disjointWith	(?C owl:disjointWith ?D) $\wedge$ (?X rdf:type ?C) $\wedge$ (?Y rdf:type ?D) $\Rightarrow$ (?X owl:differentFrom ?Y)
inverseOf	(?P owl:inverseOf ?Q) $\wedge$ (?X ?P ?Y) $\Rightarrow$ (?Y ?Q ?X)

Fonte: WANG et al. (2004)

Estas regras de inferência permitem realizar raciocínios sobre a ontologia do contexto, tornando possível o exemplo descrito na subseção 2.5.1. A Figura 2 ilustra este processo. O

analisador recebe como entrada o contexto explícito nos dados e, através do conjunto de regras, infere o contexto implícito.

Figura 2: Exemplo de inferência do contexto através de OWL

INPUT	DL Reasoning Rules	(?P rdf:type owl:TransitiveProperty) $\wedge$ (?A ?P ?B) $\wedge$ (?B ?P ?C) $\Rightarrow$ (?A ?P ?C) (?P owl:inverseOf ?Q) $\wedge$ (?X ?P ?Y) $\Rightarrow$ (?Y ?Q ?X)
	Explicit Context	<owl:ObjectProperty rdf:ID="locatedIn"> <rdf:type="owlTransitiveProperty"/> <owl:inverseOf rdf:resource="#contains"/> </owl:ObjectProperty> <Person rdf:ID="Wang"> <locatedIn rdf:resource="#Bedroom"/> </Person > < Room rdf:ID="Bedroom"> < locatedIn rdf:resource="#Home"/> </ Room>
OUTPUT	Implicit Context	<Person rdf:ID="Wang"> <locatedIn rdf:resource="#Home"/> </Person > <Building rdf:ID="Home"> < contains rdf:resource="#Bedroom"/> < contains rdf:resource="#Wang"/> </Building> <Room rdf:ID="Bedroom"> < contains rdf:resource="#Wang"/> </Room>

Fonte: WANG et al. (2004)

Diversos analisadores foram desenvolvidos com o objetivo de realizar inferências em ontologias descritas em OWL. Por exemplo o trabalho de SIRIN et al. (2007) com o *Pellet*[1], um verificador consistência e sintaxe de documentos OWL. E os raciocinadores de lógica descritiva, *FaCT++*[2] (TSARKOV; HORROCKS, 2006), e *Hermit*[3] (MOTIK; SHEARER; HORROCKS, 2009).

Apesar de não afetar diretamente a análise de similaridade, a capacidade de inferir novos relacionamentos permite expandir o conhecimento sobre um domínio, descrito em uma ontologia. Conforme é detalhado no Capítulo 4, esta é uma capacidade útil para a fase de pré-processamento, onde se busca um refinamento dos dados de entrada, aperfeiçoando a análise em situações mais complexas, ou onde os dados estejam incompletos.

2.5 Representação de dados contextuais

A maneira como os dados contextuais serão representados internamente no *framework* é de vital importância para a qualidade da análise de similaridade. Apesar da natureza heterogênea destes dados, é necessário haver uma certa organização, que permita ao menos identificar em qual categoria de dados contextuais (DEY; ABOWD; SALBER, 2001) cada informação se enquadra. Esta informação é necessária para que se possa identificar corretamente qual o tipo de

[1] http://www.mindswap.org/2003/pellet/download.shtml
[2] http://owl.man.ac.uk/factplusplus
[3] http://www.hermit-reasoner.com/

métrica de similaridade que deve ser aplicada em cada caso.

Por exemplo, o tratamento para a localização física da entidade é diferente do tratamento a ser aplicado em dados descritivos da situação, que por sua vez é completamente diferente do tratamento para os dados temporais. Portanto é necessário estabelecer um formalismo para os dados contextuais repassados ao *framework*, a fim de evitar interpretações errôneas.

Segundo VANATHI; RHYMEND UTHARIARAJ (2009) o método mais simples para mapear contextos é o modelo 'chave-valor', onde a informação é mapeada através de uma chave que indexa o valor de um determinado atributo. Neste modelo o comando para, por exemplo, marcar o 'quarto 109' como 'localização' poderia ser *'set LOC=ROOM=109'*. Porém este modelo não é eficiente para estruturas mais avançadas. Os autores descrevem outras abordagens, como os modelos baseados em lógica, UML, orientados a objetos e baseados em linguagens de marcação como SGML.

2.5.1 Ontologias para descrição de contextos

A representação de contextos em ontologias traz a vantagem de se mapear a semântica dos dados contextuais, permitindo a descoberta de novos conhecimentos sobre o domínio do contexto. Para ilustrar esta capacidade WANG et al. (2004) utilizam o exemplo de um *smartphone* que pode adaptar seu funcionamento de forma pró-ativa em função do contexto do usuário. Assim, quando o usuário estivesse dormindo ou tomando banho todas as chamadas seriam automaticamente encaminhadas para a caixa postal. Se o usuário estivesse assistindo televisão, o volume do aparelho seria automaticamente aumentado. Já se o usuário estivesse jantando com a família o aparelho seria automaticamente colocado no modo silencioso. Os autores classificam estas informações como *high-level context*, por estarem vinculadas diretamente às atividades do usuário. O problema é que não é possível obter estas informações diretamente dos sensores, já que estes apenas conseguem fornecer dados classificados como *low-level context*, como por exemplo: a localização do usuário, se a televisão está ligada ou não, quem mais está presente, e o horário. Porém se estes dados estiverem mapeados em uma ontologia, será possível tirar proveito de mecanismos de inferência para se expandir o conhecimento sobre o contexto. Por exemplo, se a localização do usuário for a sala de estar e a televisão estiver ligada, é possível inferir que o usuário está assistindo televisão. Ou, se o usuário estiver no quarto e o horário for o horário habitual de dormir pode-se inferir que o usuário estará dormindo.

Diversas ontologias vêm sendo propostas para descrição de contextos. QIN; SHI; SUO (2007) e CHEN; NUGENT; WANG (2012), por exemplo, propõem ontologias com o propósito de formalizar as entidades contextuais em espaços inteligentes. Na ontologia desenvolvida por QIN; SHI; SUO (2007), os principais conceitos mapeados são: Usuário, Localização, Tempo, Atividade, Serviço, Ambiente e Plataforma. Já a ontologia desenvolvida por CHEN; NUGENT; WANG (2012) classifica as informações contextuais em *Spatial Context*, com informações sobre a localização; *Event Context*, com informações sobre atividades e mudanças de estados

como por exemplo, a abertura de portas; e *Enviromental Context* com dados do ambiente como temperatura e umidade do ar.

O trabalho de VANATHI; RHYMEND UTHARIARAJ (2009) também estudou as abordagens para modelagem de contextos em ontologias. Foram estudadas as ontologias *Standard Ontology for Ubiquitous and Pervasive Applications - SOUPA* (CHEN; FININ; JOSHI, 2003), Gaia (RANGANATHAN et al., 2003), *Global Smart Space - GLOSS* (COUTAZ et al., 2003)[4], *Aspect-Scale-Context - ASC* (STRANG; LINNHOFF-POPIEN; FRANK, 2003) e *CONtext Ontology - CONON* (GU et al., 2004). A Figura 3 ilustra o comparativo feito pelos autores entre estas ontologias.

Figura 3: Comparação relação aos princípios de projetos de ontologias

Ontology Based Models	CoBrA SOUPA	Gaia	GLOSS	ASC CoOL	SOCAM CONON
Coherence	√				
Clarity	√			√	
Extensibility	√	√	√	√	√
Ontological Commitment	√		√		√
Orthogonality	√		√		
Encoding Bias	√	√	√		√

Fonte: VANATHI; RHYMEND UTHARIARAJ (2009)

Apesar de neste comparativo a ontologia SOUPA ter se destacado, alguns aspectos adicionais precisam ser considerados. Segundo NAUDET (2011), o desenvolvimento de ontologias para modelagem de contextos possui três abordagens:

1. ***User-Centered***, caracterizadas por um conceito central chamado Usuário ou Pessoa, ligado aos diversos elementos que caracterizam um contexto ou situação;
2. **Definição do contexto de forma geral**, onde não há um conceito central focado em um único elemento do contexto. Para o autor esta abordagem divide-se em dois subitens:
   (a) **Agregação de ontologias**, onde ontologias diferentes, cada uma dedicada a 1 ou alguns elementos do contexto, são agrupadas para criar uma nova ontologia. Por exemplo, SOUPA e Ontonym;
   (b) **Definição explícita do Contexto**, onde é estabelecido a entidade Contexto, ou equivalente, formalizando as entidades de interesse como subclasses desta entidade. Um exemplo desta abordagem é a ontologia CONON.

Porém, mesmo as ontologias CONON e SOUPA ainda são centralizadas no conceito de usuário, segundo o autor. Em aplicações que utilizem informações sobre o contexto, mas que não estejam diretamente vinculadas a atividades de usuários, é necessário uma ontologia capaz de lidar com entidades genéricas. O autor propõe então a *Generic Context Ontology*, uma

[4]Uma versão mais recente desta ontologia (2010) está disponível em: http://arxiv.org/abs/1006.5661

ontologia genérica com definição explícita do conceito de contexto e mais próxima ao conceito de contexto definido por DEY; ABOWD; SALBER (2001). A única diferença para o conceito proposto por Dey, é que Naudet não considera que o ambiente seja um sinônimo de contexto, uma vez que este conceito corresponde a informações sobre elementos que estão ao redor de um sistema e que possuem influência sobre este.

Por fim, a *UbisWorld* (HECKMANN, 2003) é uma ontologia que estende a *Block Worlds* (SLANEY; THIéBAUX, 2001) e o *Context Toolkit* (DEY; ABOWD; SALBER, 2001). Pode ser definida como uma coleção de conceitos e modelos para localização, tempo, interação e situação que foram preparados para representação na forma de uma ontologia.

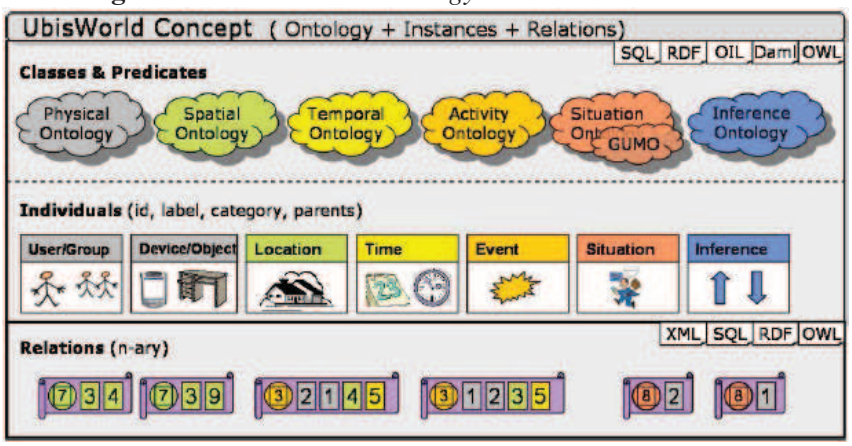
Esta ontologia pode ser utilizada para representar partes do mundo real como um escritório, uma loja, um museu ou um aeroporto. Ela representa pessoas, objetos e localizações, assim como tempo, eventos e suas propriedades e características. A Figura 4 ilustra conceitualmente esta ontologia. Os conceitos e predicados são descritos por 6 ontologias aditivas:

1. ***Physical Ontology***: Descreve os elementos físicos que são relevantes para a Computação Ubíqua e modelagem de usuários. Os elementos podem ser qualquer coisa que exista fisicamente, como pessoas, objetos e dispositivos;
2. ***Spatial Ontology***: Os elementos físicos estarão sempre localizados em um ponto específico do espaço, enquanto o usuário se move por ambientes de Computação Ubíqua. Esta ontologia mapeia conceitos como 'Países', 'Regiões', 'Cidades', 'Ruas', 'Prédios', 'Salas' e 'Parte de uma sala'. HECKMANN (2005) também descreve o conceito 'Object Level', para indicar os objetos que estão sendo manuseados pelo usuário;
3. ***Temporal Ontology***: Descreve conceitos temporais como 'data' e 'hora', bem como intervalos de tempo;
4. ***Activity Ontology***: O objetivo desta ontologia é descrever mudanças que ocorrem no mundo real. Descreve conceitos como 'evento', 'pegar', 'colocar' e 'alterar';
5. ***Situation Ontology***: Descreve atributos, parâmetros e propriedades sobre usuários, sistemas, localizações ou atividades. Exemplos de conceitos desta ontologia são 'nível de ruído', 'condições climáticas' e 'iluminação' para instâncias de localizações e 'pressão sanguínea' e 'interesses' para pessoas. A ontologia *General User Model Ontology* (GUMO) pode ser considerada como uma parte da *Situation Ontology*;
6. ***Inference Ontology***: Define regras ou processos de inferência em ambientes inteligentes.;

2.5.2 *SituationML*

Diversos autores têm proposto métodos para modelar as informações contextuais. Como por exemplo o projeto *Doppelgänger* (ORWANT, 1994), um sistema genérico de modelagem de usuários que coleta dados, realiza inferencias e disponibiliza os resultados para aplicações,

Figura 4: *UbisWorld = Ontology + Instances + Relations*



Fonte: HECKMANN (2005)

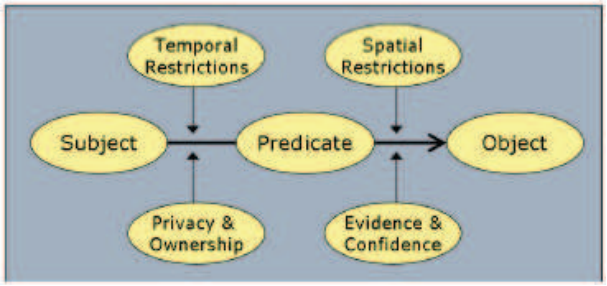
e a *Human Markup Language* (*HumanML*), uma especificação desenvolvida pela *OASIS* para descrever contextualmente informações físicas, culturais e sociais (OASIS - Organization for the Advancement of Structured Information Standards, 2013).

No entanto, segundo ROSA (2013), a *SituationML* (HECKMANN, 2005) é considerada a abordagem mais completa para a representação de contextos de entidades, considerando-se critérios como o uso de ontologias e a separação entre sintaxe e semântica.

HECKMANN (2005) propõe um modelo que separa a sintaxe da descrição de dados contextuais, composta pela linguagem *SituationML* e pela linguagem de consulta *SituationQL*, da descrição dos relacionamentos semânticos entre elementos do contexto, composta pela ontologia *UbisWorld*.

A *SituationML*, também chamada de *UserML* e *ContextML*, baseia-se na estrutura *Subject* $\rightarrow$ *Predicate* $\rightarrow$ *Object* utilizada para descrições de recursos em RDF. Segundo (HECKMANN, 2005) o objetivo da *SituationML* é permitir a escrita de sentenças mais abrangentes sobre dados contextuais, mas mantendo os dados estruturados. A Figura 5 ilustra a tupla da SituationML como uma extensão da tupla RDF, sobre a qual são acrescentadas restrições espaço-temporais e metadados para privacidade e descrições de evidências e grau de confiança.

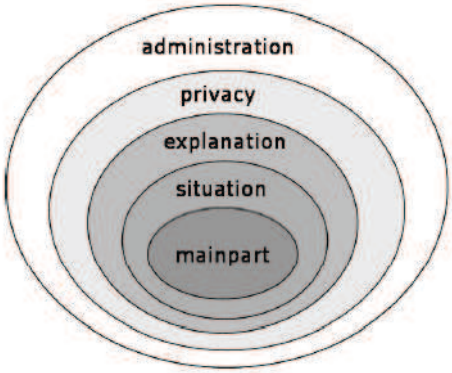
Figura 5: A tripla RDF composta por Sujeito, Predicado e Objeto. Combinada com restrições de espaço e tempo e metadados para privacidade e evidências



Fonte: HECKMANN (2005)

A estrutura formada por esta extensão do RDF é chamada de *SituationalStatement* por HECKMANN (2005). Conceitualmente, um *SituationalStatement* é composto por uma estrutura de camadas demonstrada na Figura 6.

Figura 6: Camadas que compõem um *SituationalStatement*



Fonte: HECKMANN (2005)

A camada mais interna é chamada de *MainPart* e corresponde a estrutura *Subject* $\rightarrow$ *Predicate* $\rightarrow$ *Object* definida no RDF, com a diferença de que o atributo *Predicate* foi dividido em 3 atributos: *Auxiliary*, *Predicate* e *Range*. O atributo *Range* determina a faixa de valores possíveis para o *Object*. Já o atributo *Auxiliary* permite detalhar o significado do *Predicate* e facilita a integração da SituationML com uma ontologia. O autor ilustra a utilização do atributo *Auxiliary* através de três exemplos de sentenças:

- *Peter is currently teaching;*
- *Peter likes teaching;*
- *Peter knows a lot about teaching.*

Estas três sentenças possuem o mesmo *Subject* (*Peter*) e *Predicate* (*teaching)*, mas a interpretação correta do seu significado depende de uma informação adicional que, em um *SituationalStatement*, é dada pelo atributo *Auxiliary*: *'is currently'*, *'likes'* e *'knows'*.

A segunda camada do *SituationalStatement* é chamada de *Situation Box* e é responsável por armazenar a localização e a informação temporal do contexto descrito na *MainPart*. É composta pelos atributos:

- ***Start***: Indica o momento em que a sentença descrita pelo *SituationalStatement* tornou-se válida;
- ***End***: Indica o momento em que a sentença deixou de ser válida;
- ***Durability***: Uma descrição qualitativa do período de tempo estimado em que a sentença será válida. Por exemplo, se o predicado for 'batimentos cardíacos', a durabilidade esperada para que o valor descrito em *'object'* se altere é de 'segundos';

- ***Location***: Descreve de forma qualitativa a posição ocupada pela entidade no espaço. Por exemplo, ‘casa’ ou ‘escritório’;
- ***Position***: Descreve de forma quantitativa a posição ocupada pela entidade no espaço. Por exemplo, ‘Latitude e Longitude’, ou ‘Coordenadas XYZ’.

A terceira camada é chamada de *Explanation Box* e contém meta-atributos sobre os dados contidos na primeira e na segunda camadas. Os dados presentes nesta camada permitem uma melhor compreensão da situação, além de permitir a aplicação de funcionalidades de resolução de conflitos e controle de permissões. É composta pelos seguintes atributos:

- ***Source***: Contém a origem dos dados descritos na *MainPart*, que pode ser o local onde a informação está armazenada ou o sensor que coletou o valor descrito em *object*;
- ***Creator***: Contém o nome do sistema ou da pessoa que foi responsável pela criação da sentença descrita na *SituationalStatement*;
- ***Method***: Especifica ‘como’ a informação descrita na *MainPart* foi obtida;
- ***Evidence***: Contém uma lista de evidências que dão suporte à validade da sentença;
- ***Confidence***: Um valor numérico que indica o nível de confiança que o *Creator* têm na validade da sentença.

A quarta camada é chamada de *Privacy Box* e controla os direitos de distribuição da informação descrita na sentença. É composta pelos atributos:

- ***key***: Permite anexar a chave de criptografia para bloquear o acesso não autorizado a informação descrita nesta sentença;
- ***Owner***: Identifica a pessoa ou o sistema que gerencia o controle de acesso aos dados descritos nesta sentença;
- ***Access***: Estabelece o nível de acesso aos dados. Por exemplo, ‘Público’, ‘Amigos’ e ‘Privado’;
- ***Purpose***: Restringe o uso que pode ser feito com a informação contida nesta sentença. Por exemplo, ‘Comercial’ ou ‘Pesquisa’;
- ***Retention***: Indica o período de tempo que a sentença pode ser mantida e utilizada.

A quinta camada é chamada de *Administration Box*. Trata-se da camada mais externa do *SituationalStatement* e tem como objetivo a organização dos dados em grandes conjuntos de *SituationalStatement’s*. É composta pelos atributos:

- ***Id***: Identificador único local da sentença, normalmente a chave primária do registro em um banco de dados;
- ***Unique***: Identificador único global da sentença. Trata-se de um identificador único da sentença em toda a internet. Pode ser representado por uma URI ou GUID;

- ***Replaces***: Permite que sentenças armazenadas previamente sejam alteradas por novas sentenças;
- ***Group***: Permite criar grupos para discriminar as sentenças armazenadas;
- ***Notes***: Permite que sejam acrescentadas informações adicionais, não estruturadas.

A Figura 7 exemplifica a utilização destes atributos. Percebe-se que a definição das camadas é apenas conceitual e não faz parte da linguagem expressa em RDF.

Figura 7: Formalização de um *SituationalStatement*

```
<rdf:RDF
    xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax#
    xmlns:st=http://www.u2m.org/2003/situation#
    xml:base=http://www.u2m.org/statements>
    <rdf:Description rdf:ID=a22>
        <st:subject> a1 </st:subject>
        <st:auxiliary> a2 </st:auxiliary>
        <st:predicate> a3 </st:predicate>
        <st:range> a4 </st:range>
        <st:object> a5 </st:object>
        <st:start> a6 </st:start>
        <st:end> a7 </st:end>
        <st:durability> a8 </st:durability>
        <st:location> a9 </st:location>
        <st:position> a10 </st:position>
        <st:source> a11 </st:source>
        <st:creator> a12 </st:creator>
        <st:method> a13 </st:method>
        <st:evidence> a14 </st:evidence>
        <st:confidence> a15 </st:confidence>
        <st:key> a16 </st:key>
        <st:owner> a17 </st:owner>
        <st:access> a18 </st:access>
        <st:purpose> a19 </st:purpose>
        <st:retention> a20 </st:retention>
        <st:id> a21 </st:id>
        <st:unique> a22 </st:unique>
        <st:replaces> a23 </st:replaces>
        <st:group> a24 </st:group>
        <st:notes> a25 </st:notes>
    </rdf:Description>
</rdf:RDF>
```

Fonte: HECKMANN (2005)

A estrutura de atributos definida em um *SituationalStatement* é compatível com a categorização proposta por DEY; ABOWD; SALBER (2001) para classificação de dados contextuais. A Tabela 2 relaciona cada categoria de dados contextuais com os atributos da *SituationML* que a descreve.

Tabela 2: Categorização dos dados contextuais descritos na *SituationML*

Categoria (DEY; ABOWD; SALBER, 2001)	***SituationML*** (HECKMANN, 2005)
Identidade	*Subject*
Situação	*Auxiliary*, *Predicate*, *Range* e *Object*. Auxiliado pelos atributos *Creator* e *Source*.
Tempo	*Start*, *End* e *Durability*
Localização	*Position* e *Location*

Fonte: Elaborado pelo autor

A categoria **identidade** corresponde ao atributo *Subject*, da *SituationML*. A **situação** é descrita pelos atributos *Auxiliary*, *Predicate*, *Range* e *Object*. Em alguns casos os atributos *Creator* e *Source* poderão ser utilizados para resolução de conflitos das variáveis que descrevem a situação. Por exemplo, caso o *subject* seja um reator nuclear poderão haver dois predicados

'temperatura'. Um para representar a temperatura do núcleo e outro para a temperatura externa. Neste caso, o atributo *Source* permitiria identificar qual sensor fez a leitura da temperatura, permitindo assim interpretar corretamente a qual variável da situação o *predicate* se refere. A categoria **tempo** é descrita pelos atributos *Start*, *End* e *Durability*. Por fim, a categoria **localização** é definida pelos atributos *Position* e *Location*. Percebe-se que o autor tomou o cuidado de distinguir a posição física do usuário, dada por sua latitude e longitude ou por um sistema de coordenadas cartesiano, da sua localização contextual, dada por uma descrição do local onde a entidade se encontra, como por exemplo: casa ou escritório.

Um *SituationalStatement* somente é capaz de expressar uma única sentença sobre uma entidade. Para uma descrição completa de um contexto é necessário um conjunto de *SituationalStatement's* denominado por HECKMANN (2005) como *SituationReport*. A Figura 8 ilustra um *SituationReport* descrevendo um cenário hipotético de um passageiro em um aeroporto. O *SituationReport* deste exemplo é composto por três *SituationalStatement's*, onde os dois primeiros descrevem duas sentenças sobre o contexto da entidade *'Peter'*. No primeiro, a propriedade *'timePressure'* da 'situação' está com o valor *'High'*. No segundo, a propriedade *'walkingSpeed'* está com o valor *'fast'*. A combinação destas sentenças permite descrever que 'o usuário *Peter* está com pressa, pois está atrasado para algum compromisso com hora marcada'. O terceiro *SituationalStatement* permite identificar qual é este compromisso. Trata-se da situação do vôo LH225 (entidade) cujo embarque se encerra em 10 minutos.

Figura 8: Um *SituationReport* com três *SituationalStatement* descrevendo a situação de duas entidades.

Situational Statement / Box		Situational Statement / Box		Situational Statement / Box	
Mainpart		**Mainpart**		**Mainpart**	
Subject	= Peter	Subject	= Peter	Subject	= Flight LH225
Auxiliary	= hasProperty	Auxiliary	= hasProperty	Auxiliary	= hasPlan
Predicate	= timePressure	Predicate	= walkingSpeed	Predicate	= boarding
Range	= low-medium-high	Range	= slow-medium-fast	Range	= time
Object	= high	Object	= fast	Object	= in 10 minutes
Situation		**Situation**		**Situation**	
Start	= 2003-04-16T19:15	Start	= 2003-04-16T19:14	Start	= 2003-04-16T19:14
End	= ?	End	= ?	End	= ?
Durability	= few minutes	Durability	= few minutes	Durability	= few minutes
Location	= airport.dutyfree	Location	= airport.dutyfree	Location	= airport.gate23
Position	= x34-y22-z15	Position	= x32.y23.z15	Position	= ?
Explanation		**Explanation**		**Explanation**	
Source	= peter.repository	Source	= sensor.repository	Source	= airport.repository
Creator	= airport.inference	Creator	= sensor.PW	Creator	= airport.inference
Method	= deduction13	Method	= Bayes	Method	= deduction13
Evidence	= id2, id3	Evidence	= LowLevelData	Evidence	= fight-system
Confidence	= most-probably	Confidence	= 0.8	Confidence	= 0.6
Privacy		**Privacy**		**Privacy**	
Key	= ?	Key	= ?	Key	= ********
Owner	= Peter	Owner	= Peter	Owner	= Airport
Access	= friends-only	Access	= friends-only	Access	= public
Purpose	= research	Purpose	= research	Purpose	= commercial
Retention	= 1 day	Retention	= 1 week	Retention	= 1 month
Administration		**Administration**		**Administration**	
id	= 1	id	= 2	id	= 3
unique	= u2m.org#154123	unique	= u2m.org#154124	unique	= u2m.org#154125
replaces	= ?	replaces	= u2m.org#154109	replaces	= u2m.org#152148
group	= UserModel	group	= UserModel	group	= ContextModel
notes	= ;-(	notes	= ;-|	notes	= ;-)

Fonte: HECKMANN (2005)

A especificação da *SituationML* permite representar qualquer tipo de contexto e abrange

todas as categorias de dados contextuais sugeridas por DEY; ABOWD; SALBER (2001). Isto justifica sua escolha como o modelo conceitual para representação interna das sequências de contextos a ser utilizada pelo *SIMCOP*.

2.6 Séries Temporais

Séries temporais são conjuntos de observações ordenadas cronologicamente (FU, 2011). Segundo ESLING; AGON (2012), em quase todas as áreas da ciência há a necessidade de se realizar medições de variáveis ao longo de um período de tempo, o que leva a criação destes conjuntos de dados ordenados.

Por exemplo na ciência econômica, segundo HAMILTON (1994), as séries temporais são utilizadas em econometria já que boa parte da pesquisa nesta área consiste em modelar o comportamento de variáveis econômicas ao longo do tempo. Os modelos que descrevem as séries temporais em geral constituem-se de equações diferenciais, que o autor define como expressões que relacionam uma variável y_t com seus valores prévios $(y_{t-1}, y_{t-2}, ..., y_{t-n})$.

Um exemplo é a função de *Goldfelds's*, que estima a demanda de dinheiro pelos Estados Unidos: $m_t = 0.27 + 0.72m_{t-1} + 0.19I_t - 0.045r_{bt} - 0.019r_{ct}$. O modelo de *Goldfeld's* relaciona a quantidade de dinheiro real em poder do público (m_t) com os rendimentos reais globais (I_t), as taxas de juros em contas bancárias (r_{bt}) e as taxas de juros em papéis comerciais (r_{ct}). Percebe-se que o valor da variável m em um momento t depende parcialmente do valor da variável m no momento anterior (m_{t-1}). Esta é uma equação diferencial de primeira ordem, já que somente o primeiro período anterior $(_{t-1})$ é utilizado na equação (HAMILTON, 1994).

Uma série temporal é denominada como Determinística quando puder ser descrita através de uma função como a de *Goldfeld's*. Caso haja um componente aleatório ε nesta função $(y_t = f(_t, \varepsilon))$, a série é denominada de Estocástica (BUENO, 2008). Além destas definições, AIUBE (2007) ainda classifica as séries temporais em Discretas, quando o conjunto de dados for finito ou infinito enumerável, Contínuas quando o conjunto for infinito não enumerável, Multivariada quando a série for representada por um vetor e Multidimensional quando $_t$ assume um valor maior do que 1.

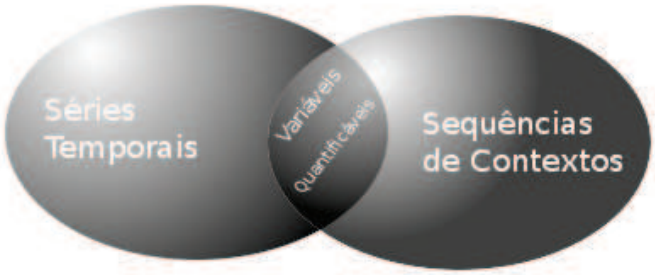
Conforme SHIKIDA; MARGARIDO (2009) uma série temporal possui geralmente os seguintes componentes:

- **Ciclos**, caracterizados por longas ondas, mais ou menos regulares, em torno de uma linha de tendência;
- **Tendencias**, utilizadas para avaliar o comportamento de longo prazo das variáveis em análise. Pode ser determinística ou estocástica;
- **Sazonalidade**, utilizado para avaliar os padrões regulares da série, afetados por eventos periódicos como temperatura, safra ou entressafra, entre outros;

- **Componente Aleatório**, constitui-se de valores residuais que não podem ser explicados por um dos três componentes anteriores.

Considerando-se a definição de série temporal dada por ESLING; AGON (2012) e FU (2011), percebe-se que as sequências de contextos visitados por uma entidade podem ser descritas como uma série temporal, desde que estejam cronologicamente ordenadas e os aspectos relevantes para a análise de similaridade entre as sequências puderem ser modelados com funções matemáticas. Isto restringe a análise de similaridade às variáveis numéricas que descrevam a situação da entidade em um momento $_t$, como por exemplo temperatura, velocidade, batimentos cardíacos, rotação, gastos, volume de vendas, e assim por diante. A Figura 9 ilustra esta correlação.

Figura 9: Relação entre séries temporais e sequências de contextos



Fonte: Elaborado pelo autor

2.7 Similaridade

Na área de descoberta de conhecimento em bases de dados (*KDD*) a tarefa de clusterização é a que possui uma relação mais próxima com o conceito de similaridade. Segundo GOLDSCHMIDT; PASSOS (2005) a clusterização, ou agrupamento, têm como objetivo particionar os registros de uma base de dados em subconjuntos chamados *clusters*, de tal forma que os elementos que compõem um *cluster* compartilhem um conjunto de propriedades que permitam distingui-los dos demais *clusters*. O resultado ótimo desta tarefa consiste em obter a maior similaridade *intra-cluster* e a maior distância *inter-cluster*.

O primeiro passo do processo de clusterização consiste em mapear os n atributos dos objetos a serem analisados, em pontos $x_1, x_2, ..., x_n$ do espaço d-dimensional R^d. Em seguida realiza-se o processo de clusterização propriamente dito, que consiste em encontrar k pontos m_j em R^d, buscando minimizar a expressão:

$$\frac{\sum_i min_j d^2(x_i,m_j)}{n}$$

Onde $d^2(x_i, m_j)$ representa a distância entre um ponto x_i e o centróide ou média do *cluster* m_j.

Ainda segundo GOLDSCHMIDT; PASSOS (2005), os algoritmos de clusterização necessitam trabalhar sobre estruturas de dados capazes de armazenar os objetos a serem processados ou as informações sobre eles. Existem duas estruturas que normalmente são utilizadas para representar os dados a serem clusterizados:

- **Matriz de dados**: Conforme ilustrado na Figura 10, a Matriz de Dados $X = n \times p$ é construida a partir da lista de objetos a serem agrupados, onde cada linha $1, ..., n$ representa um objeto e cada coluna $1, ..., p$ o valor de um atributo do objeto no espaço p-dimensional.

Figura 10: Matriz de dados

$$X = \begin{bmatrix} x_{11} & x_{12} & x_{13} & \cdots & x_{1p} \\ x_{21} & x_{22} & x_{23} & \cdots & x_{2p} \\ x_{31} & x_{32} & x_{33} & \cdots & x_{3p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \cdots & x_{np} \end{bmatrix}$$

Fonte: GOLDSCHMIDT; PASSOS (2005)

- **Matriz de similaridade**: A Matriz de Similaridade $D = n \times n$, onde n =quantidade de objetos, ilustrada na Figura 11 é construída a partir da distância entre pares de objetos. A função $d(i, j)$ representa a medida de distância ou similaridade entre um objeto $_i$ e um objeto $_j$. Quanto mais próximo de 0 (zero) for a distância entre dois objetos, mais similares eles serão. Considerando-se que a distância entre um objeto $_i$ e um objeto

Figura 11: Matriz de similaridade

$$D = \begin{bmatrix} 0 & & & & \\ d(1,2) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & d(n,3) & \cdots & 0 \end{bmatrix}$$

Fonte: GOLDSCHMIDT; PASSOS (2005)

$_j$ é igual a distância entre um objeto $_j$ e um objeto $_i$, conclui-se que não é necessário armazenar todas as distâncias na matriz.

As métricas de distância podem ser utilizadas para realizar dois tipos de consultas em bancos de dados, segundo SKOPAL; BUSTOS (2011):

1. ***Range query***: Uma *range query* (q, r) em um banco de dados $\mathbb{S}$ retorna todos os objetos cuja distância em relação ao objeto q seja menor ou igual a r;

2. ***k nearest-neighbors query (kNN)***: Retorna os k objetos em $\mathbb{S}$ mais próximos a q.

O conceito de distância como o inverso da similaridade é utilizado também por outros autores. XUECHENG (1992), por exemplo, define a distância como uma medida da diferença entre dois conjuntos difusos (*fuzzy sets*), e a similaridade como uma medida da semelhança entre estes conjuntos.

2.7.1 Medidas de similaridade métricas

Segundo CHA (2007), do ponto de vista matemático e científico a distância ou dissimilaridade é um valor que expressa o quão distante dois objetos estão. Medidas quantitativas são em geral chamadas de métricas, enquanto outras medidas não-métricas são as vezes chamadas de divergências. Já a similaridade ou proximidade entre objetos é, em alguns trabalhos, chamada de coeficiente de similaridade. O autor realizou um estudo comparativo de diversas métricas de distâncias utilizando *PDF - Probability Density Function* (PARZEN, 1962) como forma de representação de conjuntos de elementos cujo universo de valores é discreto e finito, e agrupando-as em relação a sua similaridade sintática:

1. L_p *Minkowski family*
   O matemático grego Euclides estabeleceu que a menor distância entre dois pontos é uma reta. Portanto a equação (1) apresentada na Figura 12 é conhecida como Distância Euclidiana. No século 19 Hermann Minkowski apresentou a métrica de distância conhecida como *City Block* ou *Manhattan Distance* (2). Hermann também generalizou a Distância Euclidiana e a *Manhattan Distance* através da equação (3). Quando p tende ao infinito é possível derivar a equação (4), chamada de *Chessboard Distance in 2D* ou *Chebyshev*.

Figura 12: Métricas de distância da família L_p

1. Euclidean L_2	$d_{Euc} = \sqrt{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^2}$	(1)
2. City block L_1	$d_{CB} = \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$	(2)
3. Minkowski L_p	$d_{Mk} = \sqrt[p]{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert^p}$	(3)
4. Chebyshev L_∞	$d_{Cheb} = \max_i \lvert P_i - Q_i \rvert$	(4)

Fonte: CHA (2007)

Segundo JUNIOR (2012), nas métricas de distância da família ou norma L_p, cada sequência é considerada um ponto em um espaço n-dimensional. De acordo com o valor de p obtêm-se diferentes medidas, cada uma com um comportamento específico. Para $p = 1$ é definido um espaço geométrico no qual todos os pontos possuem o mesmo valor de soma das suas diferenças absolutas. Para $p = 2$ é definido um espaço geométrico no qual todos os pontos estão equidistantes do centro. Para $p = 3$ é definido um espaço geométrico

quadrado de cantos arredondados. E para $p = \infty$ é definido um espaço quadricular onde a distância entre dois pontos é dada pela maior das suas diferenças.

2. L_1 *family*

   A equação (5) apresentada na Figura 13 é largamente utilizada em ecologia e é conhecida como *Sørensen Distance* ou *Bray-Curtis*. Corresponde a $\frac{L_1}{2}$. A *Gower distance* demonstrada nas equações (6) e (7) escala o espaço vetorial dentro do espaço normalizado e em seguida aplica L_1. Caso as sequências já estejam normalizadas, a *Gower Distance* equivale a $\frac{L_1}{d}$. Outras métricas desta família que não são proporcionais a L_1 são a *Soergel Distance* (8) e a *Kulczynski Distance* (9). A *Canberra Distance* (10) se assemelha a *Sørensen Distance*, porém normaliza a diferença absoluta a nível individual, o que a torna muito sensível a pequenas variações próximas a 0 (zero). A equação (11) atribuida a *Lorentzian*, também considera a diferença absoluta e aplica o logaritmo natural.

Figura 13: Métricas de distância da família L_1

5. Sørensen	$d_{sor} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} (P_i + Q_i)}$	(5)
6. Gower	$d_{gow} = \frac{1}{d} \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{R_i}$	(6)
	$= \frac{1}{d} \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$	(7)
7. Soergel	$d_{sg} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} \max(P_i, Q_i)}$	(8)
8. Kulczynski *d*	$d_{kul} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} \min(P_i, Q_i)}$	(9)
9. Canberra	$d_{Can} = \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{P_i + Q_i}$	(10)
10. Lorentzian	$d_{Lor} = \sum_{i=1}^{d} \ln(1 + \lvert P_i - Q_i \rvert)$	(11)
* L_1 family ⊃ {Intersectoin (13), Wave Hedges (15), Czekanowski (16), Ruzicka (21), Tanimoto (23), etc}.		

Fonte: CHA (2007)

3. *Intersection family*

   A intersecção entre duas *PDF's* calculada pela equação (12) é uma métrica amplamente utilizada para calcular a similaridade, enquanto que a não-sobreposição entre duas *PDF's* calculada pela equação (13) corresponde simplesmente a $\frac{L_1}{2}$. Por este motivo as métricas apresentadas na Figura 14 pdem ser transformadas em métricas L_1 usando esta técnica, ou seja, $dx(P, Q) = 1 - sx(P, Q)$, com algumas poucas exceções. A equação (14) é chamada de *Wave Hedges* e sua forma baseada na distância L_1 é dada pela equação (15). A forma L_1 do *Czekanowski Coefficient* na equação (16) é identica a *Sørensen Distance* (5). A metade do *Czekanowski Coefficient* é chamado de *Motyka similarity* na equação (18). A equação (20) é conhecida como *Kulczynski similarity*. A equação (22) é conhecida como *Tanimoto distance* ou *Jaccard distance*. A *Soergel distance* na equação (8) é identica a

Tanimoto. A equação (21) é conhecida como *Ruzicka similarity* e pode ser definida como $1 - dTani$. Por fim, a equação (23) ilustra estas equivalências.

Figura 14: Métricas de distância da família *Intersection*

11. Intersection $s_{IS} = \sum_{i=1}^{d} \min(P_i, Q_i)$ (12)	14. Motyka $s_{Mot} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$ (18)
$d_{non-IS} = 1 - s_{IS} = \frac{1}{2} \sum_{i=1}^{d} \lvert P_i - Q_i \rvert$ (13)	$d_{Mot} = 1 - s_{Mot} = \frac{\sum_{i=1}^{d} \max(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$ (19)
12. Wave Hedges $d_{WH} = \sum_{i=1}^{d} (1 - \frac{\min(P_i, Q_i)}{\max(P_i, Q_i)})$ (14) $= \sum_{i=1}^{d} \frac{\lvert P_i - Q_i \rvert}{\max(P_i, Q_i)}$ (15)	15. Kulczynski s $s_{Kul} = \frac{1}{d_{Kul}} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}$ (20)
13. Czekanowski $s_{Cze} = \frac{2 \sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} (P_i + Q_i)}$ (16)	16. Ruzicka $s_{Ruz} = \frac{\sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} \max(P_i, Q_i)}$ (21)
$d_{Cze} = 1 - s_{Cze} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert}{\sum_{i=1}^{d} (P_i + Q_i)}$ (17)	17. Tanimoto $d_{Tani} = \frac{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - 2 \sum_{i=1}^{d} \min(P_i, Q_i)}{\sum_{i=1}^{d} P_i + \sum_{i=1}^{d} Q_i - \sum_{i=1}^{d} \min(P_i, Q_i)}$ (22) $= \frac{\sum_{i=1}^{d} (\max(P_i, Q_i) - \min(P_i, Q_i))}{\sum_{i=1}^{d} \max(P_i, Q_i)}$ (23)

Fonte: CHA (2007)

4. *Inner Product family*

   As medidas de similaridade ilustradas na Figura 15 incorporam o produto interno $P \bullet Q$ explicitamente em suas definições. O produto interno de dois vetores é também chamado de número de correspondências ou sobreposição. O produto interno demonstrado na equação (24) é também chamado de produto escalar. A equação (25) é a média harmonica. A equação (26) é o produto interno normalizado e é chamado de coeficiente do cosseno pois calcula o ângulo entre dois vetores. A equação (27) é conhecida como *Peak-to-correlation energy* ou *PCE*. O *Jaccard coefficient*, também estudado por THORUP (2013), e demonstrado nas equações (28) e (29) é outra variação do produto interno. O coeficiente *Dice* na equações (30) e (31) é também conheecido como *Sorensen*, *Czekannowski*, *Hodgkin-Richards* ou *Morisita*.

   As métricas desta família são encontradas frequentemente nas áreas de recuperação da informação e taxonomia biologica.

5. *Fidelity family* ou *Squared-chord family*

   As métricas apresentadas na Figura 16 são baseadas no somatório das médias geométricas das *PDF's* analisadas. A equação (32) é conhecida como *Fidelity similarity*, *Bhattacharyya coefficient* ou *Hellinger affinity*. A equação 33 é chamada de *Bhattacharyya distance* e corresponde a um valor entre 0 e 1, que corresponde aos limites da probabilidade Bayesiana de erros de classificação. As outras abordagens são similares a *Bhattacharyya distance*.

6. *Squared L_2 family* ou *X^2 family*

Figura 15: Métricas de distância da família *Inner Product*

18. Inner Product	$s_{IP} = P \bullet Q = \sum_{i=1}^{d} P_i Q_i$	(24)
19. Harmonic mean	$s_{HM} = 2\sum_{i=1}^{d} \frac{P_i Q_i}{P_i + Q_i}$	(25)
20. Cosine	$s_{Cos} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sqrt{\sum_{i=1}^{d} P_i^2} \sqrt{\sum_{i=1}^{d} Q_i^2}}$	(26)
21. Kumar-Hassebrook (PCE)	$s_{Jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$	(27)
22. Jaccard	$s_{Jac} = \frac{\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$	(28)
	$d_{Jac} = 1 - s_{Jac} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2 - \sum_{i=1}^{d} P_i Q_i}$	(39)
23. Dice	$s_{Dice} = \frac{2\sum_{i=1}^{d} P_i Q_i}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$	(40)
	$d_{Dice} = 1 - s_{Dice} = \frac{\sum_{i=1}^{d} (P_i - Q_i)^2}{\sum_{i=1}^{d} P_i^2 + \sum_{i=1}^{d} Q_i^2}$	(31)

Fonte: CHA (2007)

Figura 16: Métricas de distância da família *Squared-Chord*

24. Fidelity	$s_{Fid} = \sum_{i=1}^{d} \sqrt{P_i Q_i}$	(32)
25. Bhattacharyya	$d_B = -\ln \sum_{i=1}^{d} \sqrt{P_i Q_i}$	(33)
26. Hellinger	$d_H = \sqrt{2\sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2}$	(34)
	$= 2\sqrt{1 - \sum_{i=1}^{d} \sqrt{P_i Q_i}}$	(35)
27. Matusita	$d_M = \sqrt{\sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2}$	(36)
	$= \sqrt{2 - 2\sum_{i=1}^{d} \sqrt{P_i Q_i}}$	(37)
28. Squared-chord	$d_{sqc} = \sum_{i=1}^{d} (\sqrt{P_i} - \sqrt{Q_i})^2$	(38)
$s_{sqc} = 1 - d_{sqc}$	$s_{sqc} = 2\sum_{i=1}^{d} \sqrt{P_i Q_i} - 1$	(39)

Fonte: CHA (2007)

A Figura 17 apresenta algumas medidas de distância que baseiam-se na *Squared Euclidian Distance* apresentada na equação (40).

Figura 17: Métricas de distância da família *Squared* L_2

29. Squared Euclidean	$d_{sqe} = \sum_{i=1}^{d} (P_i - Q_i)^2$	(40)
30. Pearson χ^2	$d_P(P,Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{Q_i}$	(41)
31. Neyman χ^2	$d_N(P,Q) = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i}$	(42)
32. Squared χ^2	$d_{SqChi} = \sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$	(43)
33. Probabilistic Symmetric χ^2	$d_{PChi} = 2\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{P_i + Q_i}$	(44)
34. Divergence	$d_{Div} = 2\sum_{i=1}^{d} \frac{(P_i - Q_i)^2}{(P_i + Q_i)^2}$	(45)
35. Clark	$d_{Clk} = \sqrt{\sum_{i=1}^{d} \left(\frac{\lvert P_i - Q_i \rvert}{P_i + Q_i} \right)^2}$	(46)
36. Additive Symmetric χ^2	$d_{AdChi} = \sum_{i=1}^{h} \frac{(P_i - Q_i)^2 (P_i + Q_i)}{P_i Q_i}$	(47)
* Squared L_2 family $\supset$ {Jaccard (29), Dice (31)}		

Fonte: CHA (2007)

7. *Shannon's entropy family*

   As equações (48) a (53) apresentadas na Figura 18 derivam do conceito de incerteza probabilistica ou "entropia"apresentado por Shannon: $H(P) = \sum_{i=1}^{d} P_i \times lnP_i$.

A equação (48) é chamada de *KL Divergence*, *Relative Entropy* ou *Information Deviation*. A forma simétrica da *KL Divergence* utilizando o método de adição é apresentada na equação (49) e é chamada de *Jeffreys* ou *J Divergence*. A equação (50) é chamada de *K Divergence* e sua forma simétrica, conhecida como *Topsøe Distance*, que utiliza o método de adição é apresentada na equação (51). A metade da *Topsøe Distance* é chamada de *Jensen-Shannon divergence* e utiliza o método de médias. A ideia de raio de informação como uma métrica, demonstrada na equação (53) surgiu devido a propriedade de concavidade da entropia de Shannon.

Figura 18: Métricas de distância da família *Shannon's Entropy*

37. Kullback–Leibler	$d_{KL} = \sum_{i=1}^{d} P_i \ln \frac{P_i}{Q_i}$	(48)
38. Jeffreys	$d_J = \sum_{i=1}^{d} (P_i - Q_i) \ln \frac{P_i}{Q_i}$	(49)
39. K divergence	$d_{Kdiv} = \sum_{i=1}^{d} P_i \ln \frac{2P_i}{P_i + Q_i}$	(50)
40. Topsoe	$d_{Top} = \sum_{i=1}^{d} \left(P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right)$	(51)
41. Jensen-Shannon	$d_{JS} = \frac{1}{2} \left[\sum_{i=1}^{d} P_i \ln \left(\frac{2P_i}{P_i + Q_i} \right) + \sum_{i=1}^{d} Q_i \ln \left(\frac{2Q_i}{P_i + Q_i} \right) \right]$	(52)
42. Jensen difference	$d_{JD} = \sum_{i=1}^{k} \left[\frac{P_i \ln P_i + Q_i \ln Q_i}{2} - \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2} \right) \right]$	(53)

Fonte: CHA (2007)

8. *Combinations*

A Figura 19 apresenta as medidas de distância que utilizam múltiplas métricas. A equação (54) denominada *Taneja* utiliza médias aritméticas e geométricas em conjunto com as divergências médias artiméticas e geométricas. A equação (55) utiliza as divergências médias Aritmética, Geométrica e *Simmetric* X^2. Por fim o valor médio da *City Block Distance* e da *Chebyshev Distance* é utilizado na equação (56).

Figura 19: Métricas de distância da família *Combinations*

43. Taneja	$d_{TJ} = \sum_{i=1}^{d} \left(\frac{P_i + Q_i}{2} \right) \ln \left(\frac{P_i + Q_i}{2\sqrt{P_i Q_i}} \right)$	(54)
44. Kumar-Johnson	$d_{KJ} = \sum_{i=1}^{d} \left(\frac{(P_i^2 - Q_i^2)^2}{2(P_i Q_i)^{3/2}} \right)$	(55)
45. Avg(L_1 ,L_∞)	$d_{ACC} = \frac{\sum_{i=1}^{d} \lvert P_i - Q_i \rvert + \max_i \lvert P_i - Q_i \rvert}{2}$	(56)

Fonte: CHA (2007)

2.7.2 Medidas de similaridade não-métricas

Segundo (SKOPAL; BUSTOS, 2011), a análise de similaridade em aplicações baseadas em bancos de dados durante muito tempo esteve restrita a aplicação de métricas de distância. No

entanto, devida a crescente complexidade dos dados em diversos domínios, nos ultimos anos vêm sendo desenvolvidas técnicas de análise de similaridade/distância não-métricas. Para os autores, uma métrica de distância possui as seguintes propriedades:

$$(\forall x, y, z \in \mathbb{U}):$$

$$\begin{aligned} \delta(x,y) = 0 \Leftrightarrow x = y \quad & \text{reflexividade} \\ \delta(x,y) > 0 \Leftrightarrow x \neq y \quad & \text{não-negatividade} \\ \delta(x,y) = \delta(y,x) \quad & \text{simetria} \\ \delta(x,y) + \delta(y,z) \geq \delta(x,z) \quad & \text{desigualdade triangular} \end{aligned}$$

Onde x, y, z são objetos pertencentes ao universo $\mathbb{U}$ de objetos (descritores) válidos de um domínio e $\delta : \mathbb{U} \times \mathbb{U} \rightarrow \mathbb{R}$ é a função de desigualdade entre dois objetos

Para (SKOPAL; BUSTOS, 2011) o desenvolvimento de análises não-metricas de similaridade surgiu a partir de objeções feitas a estes axiomas. Por exemplo, as Figuras 20a e 20b ilustram a alegação de que diferentes objetos podem possuir auto-similaridade diferente, o que refuta os conceitos de reflexividade e não-negatividade. Na Figura 20a, a comparação de uma imagem com ela mesma pode resultar em um valor de desigualdade maior do que 0 (zero) caso utilize-se uma métrica que avalie as partes menos similares dos dois objetos, no caso o tronco e a folha. Já na Figura 20b, a comparação de duas imagens diferentes pode resultar em um valor de desigualdade 0 (zero) caso utilize-se uma métrica que avalie somente as partes mais similares dos objetos, no caso a folha solta e a folha presa ao tronco.

Figura 20: Objeções às propriedades métricas de similaridade

Fonte: SKOPAL; BUSTOS (2011)

A Figura 20c refuta o conceito de simetria na similaridade de dois objetos. Considerando-se o objeto x como o mapa da Irlanda e o objeto y como o mapa do Reino Unido, obtem-se uma situação onde a desigualdade entre x e y é menor do que a desigualdade entre y e x já que x está contido em y, mas y não está contido em x.

Por fim, a Figura 20d ilustra um dos pontos mais atacados no que se refere a análise de similaridade baseada em métricas, segundo (SKOPAL; BUSTOS, 2011): A de que a similaridade não precisa ser transitiva. No exemplo, um homem é semelhante a um centauro e um centauro é semelhante a um cavalo, no entanto um homem é completamente diferente de um cavalo.

SKOPAL; BUSTOS (2011) descreve as seguintes medidas de similaridade não-métricas:

1. **Fractional Lp**:

$L_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Dados os vetores $x, y \in \mathbb{R}$. Trata-se de uma variação da norma L_p onde $p \in (0..1)$. Nesta situação não é possível manter a propriedade da Desigualdade Triangular.

2. **Dynamic Partial Function**

$\delta_{DPF}(x,y) = \left(\sum_{c_i \in \Delta_m} |x_i - y_i|^p\right)^{\frac{1}{p}}$

Para $p \geq 1$. Onde $c_i = |x_i - y_i|$, x_i e y_i correspondem a i-ésima coordenada dos vetores x e y, e Δ_m é o conjunto dos $_m$ menores valores de $\{c_1, ..., c_d\}(m \leq d)$. Similar a **Minkowski distance**, onde apenas alguns poucos valores de coordenadas são utilizados. Esta medida não satisfaz a propriedade da Desigualdade Triangular.

3. **Cosine Measure and Distance**

$S_{\cos} = \frac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2 \cdot \sum_{i=1}^{d} y_i^2}}$

Utilizada em situações onde deseja-se analisar a similaridade entre dois objetos através do ângulo entre os vetores que representam estes objetos, ignorando-se a magnitude destes vetores. A Distância do Cosseno é uma medida de desigualdade semi-simétrica, e portanto não-métrica, obtida através da fórmula $\delta_{\cos}(x,y) = 1 - S_{\cos}(x,y)$. Já a Distância do Ângulo é obtida através da fórmula $\delta_\theta(x,y) = \arccos(S_{\cos}(x,y))$ e é uma medida de desigualdade métrica.

4. **Kullback-Leibler Divergence**

$\delta_{KLD}(x,y) = \sum_{i=1}^{d} x_i \cdot \log\left(\frac{x_i}{y_i}\right)$

É uma medida de desigualdade entre histogramas baseada na Teoria da Informação, que mede o quão ineficiente seria, em média, codificar um histograma utilizando o outro como distribuição válida para codificação. Esta medida não satisfaz as propriedades Simetria, Não-Negatividade e Desigualdade Triangular.

5. **Jeffrey Divergence**

$\delta_{JD}(x,y) = \sum_{i=1}^{d} x_i \cdot \log\left(\frac{x_i}{\frac{x_i+y_y}{2}}\right) + y_i \cdot \log\left(\frac{y_i}{\frac{x_i+y_y}{2}}\right)$

Também baseada na Teoria da Informação, esta medida satisfaz a propriedade da Simetria, mas não satisfaz a Desiguladade Triangular.

6. χ^2 **Distance**

$\delta_{\chi^2}(x,y) = \sum_{i=1}^{d} \frac{x_i - m(i)}{m(i)}$

Onde $m(i) = \frac{x_i+y_i}{2}$, ou seja o valor médio entre x_i e y_i. Esta técnica mede se duas distribuições foram produzidas pela mesma distribuição real subjacente. Esta medida não satisfaz as propriedades Simetria, Não-Negatividade e Desigualdade Triangular.

7. **Dynamic Time Warping Distance**

$\delta_{DTW}(x, y) = min_w \left\{ \sqrt{\sum_{k=1}^{t} w_k} \right\}$

Esta é uma técnica desenvolvida para calcular a diferença entre duas séries temporais. Consiste em mapear as séries $x = \{x_1, x_2, ..., x_n\}$ e $y = \{y_1, y_2, ..., y_m\}$ em uma matriz $n \times m$ onde $M[i, j] = (x_i - y_i)^2$ e em seguida encontrar o menor caminho w através de M, considerando-se o somatório do valor das células que compõem w. Esta medida não satisfaz a propriedade da Desigualdade Triangular.

8. **Longest Common Subsequence**

   Uma sequência x é uma subsequência de uma sequência y se existir uma sequência estritamente crescente de índices i onde existe correspondência entre x_i e y_i. A **Longest Common Subsequence** é uma função de similaridade S_{LCS} que retorna o tamanho da maior subsequência entre x e y. Esta medida não satisfaz a propriedade da Desigualdade Triangular.

9. **Earth's Mover's Distance**

   $\delta_{EMD}(x, y) = min \left\{ \sum_{i=1}^{d} \sum_{j=1}^{d} c_{ij} f_{ij} \right\}$

   Onde:

   $f_{ij} \geq 0$;

   c_{ij} é o custo para transformar o valor x_i no valor de y_j;

   $\sum_{i=1}^{d} f_{ij} = y_i \quad \forall j = 1, ..., d$;

   $\sum_{i=1}^{d} f_{ij} = x_i \quad \forall i = 1, ..., d$

   Esta técnica mede a quantidade de trabalho necessário para transformar uma distribuição de valores (um vetor de características) em outra, considerando qualquer par de dimensões e não apenas dimensões equivalentes entre ambas as distribuições. Os valores de c_{ij} são também chamados de *ground distance* e se forem métricos a EMD atenderá os requisitos de uma medida métrica, caso contrário somente a propriedade da Simetria é garantida por esta medida.

10. **Hausdorff Distance Variants**

    $\delta_{HD}(A, B) = max\{h(A, B), h(B, A)\}$

    Onde:

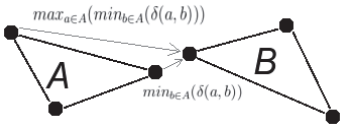
    (A, B) são conjuntos de objetos $A = \{a_1, ..., a_m\}$ e $B = \{b_1, ..., b_n\}$;

    $h(A, B)$ é uma função que retorna a maior distância entre pontos do conjunto A até o ponto mais próximo no conjunto B, ou seja, $h(A, B) = max_{a \in A}(min_{b \in A}(\delta(a, b)))$, sendo $\delta(a, b)$ uma medida de distância qualquer.

    A distância de Hausdorff mede o quão distante um conjunto está do de outro através da comparação entre pontos dos dois conjuntos, retornando a maior distância entre algum ponto do conjunto A com o ponto mais próximo de A no conjunto B, conforme ilustrado

na Figura 21, onde esta medida é utilizada para calcular a distância entre dois polígonos.

Figura 21: Distância de Hausdorff



Fonte: Adaptado de GRéGOIRE; BOUILLOT (2013)

11. **Normalized Edit Distance**

A *Edit Distance*, também conhecida como *Levenshtein Distance* mede a quantidade mínima de operações necessárias para transformar uma *string* em outra. São permitidas as operações "Inserir", "Apagar" e "Substituir". A cada operação pode ser atribuido um peso γ, composto por um número real positivo, $\gamma(a \rightarrow b)$ para substituições, $\gamma(a \rightarrow \lambda)$ para exclusões e $\gamma(\lambda \rightarrow b)$ para inserções. Um *editing path* P entre duas *strings* é um conjunto de pares ordenados (i_k, j_k) $(0 \leq k \leq m$, com $m(P)$ sendo a quantidade de operações no caminho. O caminho ponderado pelos pesos $W(P)$ é definido como:

$$W(P) = \sum_{k=1}^{m} \gamma(x_{ik-1} + 1...i_k \rightarrow y_{jk-1} + 1...j_k)$$

A versão normalizada da *edit distance* leva em conta o tamanho do *editing path* e é definida como:

$$\delta_{NED}(x, y) = min\{\hat{W}(P)\}$$

Onde:
$\hat{W}(P) = \frac{W(P)}{m(P)}$.

Esta medida não satisfaz a propriedade da Desiguladade Triangular.

12. **Sequence Alignment Distance**

A distância de alinhamento de sequências é uma generalização da *edit distance*. Em problemas de correspondência de sequências o objetivo é, dada uma sequência $y = \{y_1, y_2, ..., y_d\}$ encontrar, em um conjunto de sequências $\mathbb{S}$, um objeto $x \in \mathbb{S}$ que minimize uma função de distância δ. A função de distância δ calcula a quantidade de operações (*match*, *replace*, *delete* e *insert*) necessária para converter uma sequência y em x. O cálculo da distância é definido como:

$$\delta_{SAD}(x, y, i, j) = min \begin{cases} c(x_i, y_i) + \delta_{SAD}(x, y, i+1, j+1) \\ c(-, y_i) + \delta_{SAD}(x, y, i, j+1) \\ c(x_i, -) + \delta_{SAD}(x, y, i+1, j) \end{cases}$$

Onde:
- $c(a, b)$ é o custo da correspondência (*match*) entre a e b;
- $c(a, -)$ é o custo para inserir o caracter a;
- $c(-, b)$ é o custo para apagar o caracter b;
- i, j são os pontos iniciais das sequências x e y respectivamente.

Dependendo da função δ_{SAD} escolhida, esta medida pode violar todas ou algumas das propriedades das métricas.

13. **Combined Functions**
Ao modelar as funções de similaridade como equações ou algoritmos torna-se possível combinar diferentes medidas, através métodos de agregação como somatórios, médias, multiplicações, entre outros, criar novas medidas. Porém, segundo SKOPAL; BUSTOS (2011), mesmo que as medidas que foram combinadas sejam métricas, o resultado da analise combinada pode ser não-métrico. Por exemplo, o somatório de duas métricas será sempre uma nova métrica. Já sua multiplicação resultará em uma medida não-métrica.

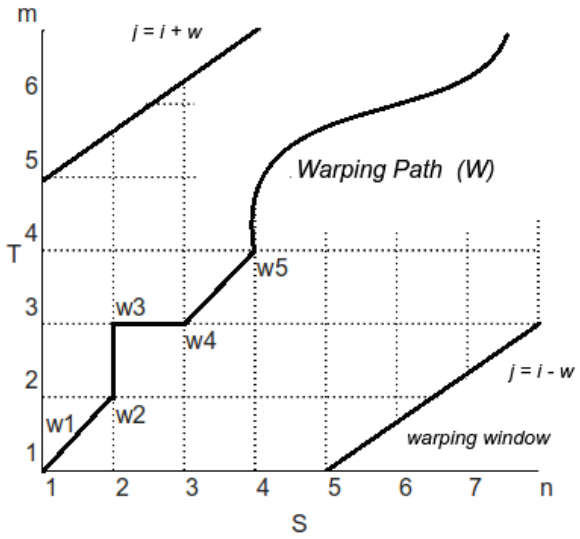
A técnica conhecida como *Dynamic Time Warping* (DTW) possui aplicações na análise de similaridade em diversos tipos de séries temporais como por exemplo a análise do histórico de preços de ações e o reconhecimento de fala, onde as ondas sonoras são mapeadas em valores ordenados cronologicamente e o reconhecimento de palavras é feito comparando-se a similaridade entre os valores recebidos com *templates* de valores previamente armazenados e que representam palavras reconhecidas (BERNDT; CLIFFORD, 1994).

Considerando-se duas séries temporais $S = \{s_1, s_2, ..., s_i, ..., s_n\}$ e $T = \{t_1, t_2, ..., t_j, ..., t_m\}$, a *DTW* alinha ambas as séries na forma de um *grid* $n \times m$, conforme ilustrado na Figura 22, onde n e m são os tamanhos das respectivas séries e cada ponto $\{i, j\}$ do *grid* corresponde a um alinhamento entre os elementos das séries. O valor de cada ponto é obtido a partir de uma medida de distância δ entre os pontos, como a magnitude da diferença ($\delta(i, j) = |i - j|$) ou o quadrado da diferença ($\delta(i, j) = (i - j)^2$). O cálculo da distância consiste utilizar técnicas de programação dinâmica para encontrar a sequência de pontos no *grid* $W = \{w_1, w_2, ..., w_k\}$ que corresponda ao menor somatório de distâncias. Para BERNDT; CLIFFORD (1994) a *DTW* é então dada por:

$$DTW(S, T) = min_w \left[\sum_{k=1}^{p} \delta(w_k) \right]$$

Recentemente outros autores vem propondo medidas de similaridade em sequências, especialmente em simlaridade de trajetórias. ZHANG et al. (2012a) propõem uma função de similaridade denominada Maximum Common Grid (MCG), que representa as trajetórias no formato $T = \{(x_i, y_i, t_i) | i = 1, 2, ..., n\}$, onde (x_i, y_i) significam a posição de um objeto qualquer em um momento t_i. O conjunto de pontos de duas trajetórias é então mapeado em uma grade $m \times n$ onde cada célula representa a posição do objeto nas coordenadas x e y. A sequência de células que compõem cada trajetória na grade é comparada e a similaridade é dada pela quantidade de células em comum entre as duas trajetórias, podendo ser calculada de três formas:

Figura 22: *Dynamic Time Warping*



Fonte: Adaptado de BERNDT; CLIFFORD (1994)

1. Similaridade Absoluta: $aSim(T_a^g, T_b^g) = |T_a^g \cap T_b^g|$;
2. Similaridade Relativa: $rSim(T_a^g, T_b^g) = \frac{|T_a^g \cap T_b^g|}{|T_a^g \cup T_b^g|}$;
3. Similaridade Parcial: $pSim(T_a^g, T_b^g) = \frac{|T_a^g \cap T_b^g|}{|T_a^g| + |T_b^g| - |T_a^g \cup T_b^g|}$;

A técnica utilizada por ZHANG et al. (2012a) para calcular a similaridade de sequências através da intersecção de conjuntos também é utilizada por outros autores. ABRAHAM; LAL (2012) utilizou o mesmo conceito para calcular a similaridade entre trajetórias de veículos. E THORUP (2013) define o coeficiente de *Jaccard* para a similaridade entre dois conjuntos A e B através da fórmula $f = \frac{|A \cap B|}{|A \cup B|}$.

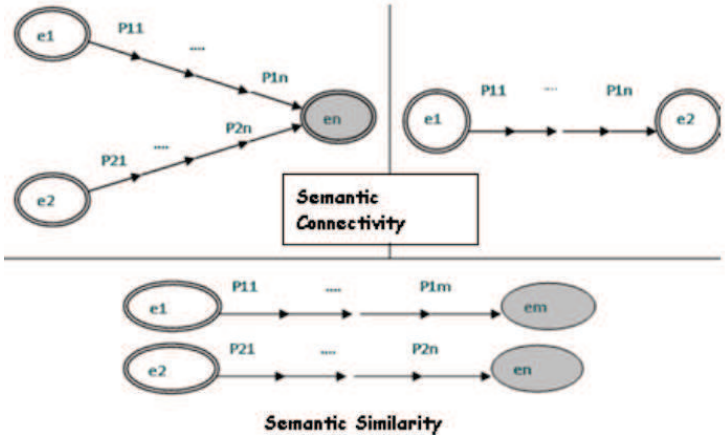
2.7.3 Similaridade semântica

Segundo SHARIATMADARI et al. (2009) existem dois tipos de análise de similaridade entre entidades descritas por uma ontologia. As entidade podem estar semanticamente conectadas caso exista pelo menos um caminho direto ou indireto ligando-as em um grafo RDF. De outra forma, as entidades podem ser semanticamente similares caso haja dois caminhos similares emanando delas. A Figura 23 ilustra estas duas formas de análise.

GAN et al. (2011) classificam os métodos para análise em três categorias: métodos baseados na estrutura de uma ontologia, métodos baseados nas informações contidas em uma ontologia e métodos que utilizam multiplas propriedades de forma híbrida. Os autores propõem um modelo denominado *DOPCA* que combina duas medidas de similaridade. A *Degrees of Overlap in Paths - DOP* e a *Depth of the Lowest Common Ancestor Node - DLCA*.

Os autores assumem que uma ontologia possui um grafo direcionado aciclico (*DAG - Directed Acyclic Graph* $G = (V, E)$ onde V corresponde ao número de vértices que, por sua vez,

Figura 23: Similaridade Semântica e Conectividade Semântica



Fonte: SHARIATMADARI et al. (2009)

representam os conceitos de uma ontologia e E corresponde ao número de arestas que, por sua vez, representam os relacionamentos entre os conceitos. Também são assumidos os seguintes axiomas, baseados na literatura, para o cálculo do valor da similaridade entre dois conceitos A e B vinculados a uma ontologia:

1. $Sim(A, B) \in \{0..1\}$;
2. $Sim(A, B) = 1$ quando $A = B$;
3. A semelhança entre A e B está relacionada aos aspectos comuns que os conceitos compartilham. Quanto mais atributos forem compartilhados entre ambos os conceitos, maior será a sua similaridade;
4. A semelhança entre A e B está relacionada a distância entre os dois termos na ontologia. Quanto mais próximo estiverem os dois termos, mais similares eles serão;
5. A similaridade semântica aumenta em razão da profundidade do nó LCA (*Lowest Common Ancestor*) dos dois termos no grafo da ontologia.

As técnicas combinadas pelos autores envolvem as seguintes equações:

1. ***DOP - Degrees of Overlap in Paths***: $Sim_{DOP}(A, B) = \frac{w_v \times |V_{A \cap B}| + w_e \times |E_{A \cap B}}{w_v \times |V_{A \cup B}| + w_e \times |E_{A \cup B}}$
   Onde $|A|$ é a cardinalidade do conjunto A, e w_v e w_e correspondem respectivamente aos pesos das arestas e dos vértices que compõem cada conjunto.
   Os autores propõem uma modificação nesta fórmula, considerando a taxa entre os pesos das arestas e dos vértices dada por $r_e = \frac{w_e}{w_V}$:
   $Sim_{DOP}(A, B) = \frac{|V_{A \cap B}| + r_e \times |E_{A \cap B}}{|V_{A \cup B}| + r_e \times |E_{A \cup B}}$

2. ***DLCA - Depth of the Lowest Common Ancestor Node***: $Sim_{DLCA}(A, B) = exp\left(-\frac{\lambda}{D_{LCA}}\right)$
   Onde $\lambda \in \{0..1\}$ é um parâmetro definido pelo usuário, e D_{LCA} é o grau do menor nó comum aos conceitos A e B.

LIU; SHAO (2010) propõem um modelo de análise de similaridade baseado no cosseno do ângulo (*cosine distance*) entre dois vetores que representem conceitos de uma ontologia. Os autores propõem a decomposição das características de um determinado conceito em um vetor, sendo que diferentes técnicas podem ser usadas para realizar esta decomposição. Um dos exemplos dados pelos autores é a criação de um vetor baseado no nome do conceito representando a taxa de ocorrências de cada letra, conforme ilustrado na Tabela 3.

Tabela 3: Decomposição de um conceito de uma ontologia em um vetor de características

Nome do Conceito	**A**	**B**	**C**	**D**	**E**
abcd	$1/4$	$1/4$	$1/4$	$1/4$	$0/4$
abde	$1/4$	$1/4$	$0/4$	$1/4$	$1/4$

Fonte: Adaptado de LIU; SHAO (2010)

Segundo LIU; SHAO (2010), a medida de similaridade baseada no cosseno mede o ângulo ou a diferença de direção entre dois vetores em um espaço vetorial. Os autores propõem que quando dois conceitos são descritos na forma de vetores de características, a similaridade entre ambos pode ser medida através da fórmula do cosseno (SKOPAL; BUSTOS, 2011). Dados dois vetores de características $x = \{1, 2, ..., n\}$ e $y = \{1, 2, ..., n\}$, o cosseno do ângulo entre os vetores no espaço n-dimensional $\mathbb{R}$, indicaria a similaridade entre os conceitos representados por estes vetores, onde $(cos = 1) \Rightarrow (\theta = 0\,^\circ)$ significaria equivalência total e $(cos = 0) \Rightarrow (\theta = 90\,^\circ)$ significaria nenhuma equivalência.

Aplicando-se a fórmula do cosseno aos vetores descritos na Tabela 3, obtém-se o valor de similaridade de 0.75:

1. $sim_{\cos} = \dfrac{\sum_{i=1}^{d} x_i y_i}{\sqrt{\sum_{i=1}^{d} x_i^2 \cdot \sum_{i=1}^{d} y_i^2}}$
2. $sim_{\cos} = \dfrac{(1/4\times1/4)+(1/4\times1/4)+(1/4\times1/4)+(1/4\times0/4)+(0/4\times1/4)}{\sqrt{1/4^2+1/4^2+1/4^2+1/4^2+0/4^2}\times\sqrt{1/4^2+1/4^2+1/4^2+0/4^2+1/4^2}}$
3. $sim_{\cos} = 0.75$

THIAGARAJAN; MANJUNATH; STUMPTNER (2008) também utilizam a fórmula do cosseno para analisar a similaridade semântica entre dois conjuntos de palavras que descrevam duas entidades. Os autores utilizam o conceito de *Bag-Of-Words (BOW)* (GHAOUI, 2012): uma lista de termos que descrevem duas entidades, onde a relevância de cada termo para a descrição da entidade é dada por um peso.

GABRILOVICH; MARKOVITCH (2007) propuseram uma extensão à distância do cosseno que analisa a similaridade semântica entre entidades utilizando os conceitos de uma ontologia descrevê-las, ao invés de utilizar uma lista de termos. Os autores denominam esta forma de representação como *Bag-Of-Concepts (BOC)*. Este estudo demonstrou que a representação em conceitos resulta em uma análise de similaridade baseada na distância do cosseno mais precisa do que a representação em BOW's.

A técnica de análise de similaridade baseada em ontologias proposta por THIAGARAJAN; MANJUNATH; STUMPTNER (2008) permite calcular a similaridade entre descrições de entidades (*Entity Description - ED*) que estejam representadas tanto na forma de conjuntos (*BOW's*) quanto de grafos (*BOC's*). O processo descrito pelos autores é composto de 6 passos:

1. Cálculo de similaridade entre duas ED's originais;
2. Caso a condição de parada tenha sido atingida, ir para o passo 5, caso contrário, executar o processo de *spread* em ambas as ED's. Os autores denominam *spread* o processo de adicionar em uma ED, os termos que possuirem relação com os termos originais;
3. Calcular a similaridade do cosseno entre as duas ED's extendidas;
4. Voltar para o passo 2;
5. Calcular a média dos valores de similaridade obtidos em todas as iterações;
6. Retornar o valor médio de similaridade.

SELVAKUMAR; SUDHA (2009) e ZHAO; WANG; DENG (2009) propõem um método para análise de similaridade entre termos de pesquisa realizados em sistemas de *e-commerce*, baseando-se na taxonomia de produtos consultados. A técnica consiste em, primeiramente, reduzir o tamanho da taxonomia base para conter apenas as classes que estão relacionadas à pesquisa, e em seguida calcular a similaridade entre dois termos através da fórmula:

$$S(T1, T2) = \frac{\sum_{i=1}^{n} [\frac{1}{m} \delta(T1, T2)]}{n}$$

Onde $T1$ e $T2$ são os dois termos em análise, n é a quantidade de atributos que foram utilizados no termo consultado, m corresponde a quantidade de valores possíveis que um atributo pode ter, e:

$$\delta(T1, T2) = \begin{cases} 1 & \text{, quando os primeiros } i \text{ atributos de T1 e T2 forem similares} \\ 0 & \text{, quando os primeiros } i \text{ atributos de T1 e T2 não forem similares} \end{cases}$$

Para THONGKRAU; LALITROJWONG (2010) a similaridade entre dois conceitos x e y também é um número real entre 0 e 1, sendo que $Sim(x, y) = 1$ somente quando $x = y$. Baseados nesta premissa, os autores afirmam que o cálculo de similaridade entre dois termos de um vocabulário pode obtido através de três maneiras:

1. *Character String Matching*, onde podem ser aplicadas técnicas como a *Edit Distance* ou a *String Matching*. Estas técnicas não são muito eficientes segundo os autores, pois palavras diferentes podem ter o mesmo significado;
2. *Similarity Algorithm bases on Synonyms Matching*, que calcula a similaridade semântica baseada em um dicionário como o *WordNet*[5] através da fórmula:

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

$$Sim_{wn}(A,B) = \frac{N(U_A \cap U_B)+\alpha}{N(U_A \cap U_B)+\alpha+\alpha\frac{|N(U_A)-N(U_B)|}{N(U_A)+N(U_B)}}$$

Onde U_A representa o conjunto de sinônimos de A, $N(U_A)$ representa o número de sinônimos de A no vocabulário, e:

$$\alpha = \begin{cases} 0 & , U_A = U_B \\ \frac{Dis_{CPI}+1}{Dis_{AB}+1} & , U_A \neq U_B. \text{ Para } Dis_{CPI} \leq Dis_{AB}, \text{ onde } Dis_{CPI} \text{ representa a distância} \\ & \text{mínima entre A e B e sua classe comum mais próxima.} \\ & \text{E } Dis_{AB} \text{ é a distância entre A e B} \end{cases}$$

3. *Similarity Algorithm based on Concept Property*. Segundo THONGKRAU; LALITROJWONG (2010) a mesma palavra pode ter significados diferentes em função do contexto onde é usada. Por exemplo a palavra inglesa *fan* pode significar tanto "ventilador" quanto "admirador". O conjunto de propriedades vinculados ao termo pode ajudar a distinguir o significado. Por exemplo, um ventilador pode possuir o conjunto de propriedades {potência, tipo, nível de ruído, ...}, já um admirador poderia possuir um conjunto de propriedades como {idade, *hobby*, ...}.

2.8 Considerações sobre o capítulo

Neste capítulo foram apresentados os conceitos teóricos que serviram de embasamento à pesquisa realizada. Inicialmente foi introduzido o conceito de aplicações sensíveis ao contexto (DEY; ABOWD; SALBER, 2001; PASCOE; RYAN; MORSE, 1998; RODDEN et al., 1998; FRANKLIN; FLASCHBART, 1998; BROWN; BOVEY; CHEN, 1997; WARD; JONES; HOPPER, 1997; SCHILIT; THEIMER, 1994) e sua importância para a área de computação ubíqua (SATYANARAYANAN, 2001; WEISER, 1991). Em seguida foram discutidos os conceitos de trilhas, sequências e históricos de contextos (LI; EICKHOFF; VRIES, 2012; SILVA et al., 2010; SMITH, 2008; DRIVER; CLARKE, 2008; SPENCE; DRIVER; CLARKE, 2005; DRIVER; CLARKE, 2004; BUSH; WANG, 1945).

Foi apresentada uma introdução às ontologias (GRUBER, 2008; HORROCKS et al., 2007) e *web* semântica (BERNERS-LEE; HENDLER; LASSILA, 2001).

Também foram discutidas as formas de representação de dados contextuais (CHEN; NUGENT; WANG, 2012; NAUDET, 2011; VANATHI; RHYMEND UTHARIARAJ, 2009; QIN; SHI; SUO, 2007; WANG et al., 2004; STRANG; LINNHOFF-POPIEN; FRANK, 2003; HECKMANN, 2003; ORWANT, 1994), e apresentada a *SituationML* (HECKMANN, 2005).

Uma introdução às séries temporais (ESLING; AGON, 2012; FU, 2011; SHIKIDA; MARGARIDO, 2009; BUENO, 2008; AIUBE, 2007; HAMILTON, 1994) foi apresentada e demonstrada a sua relação com o problema da similaridade em sequências de contextos.

Por fim, foram estudados os conceitos e métricas utilizados para a análise de similaridade (THORUP, 2013; GRéGOIRE; BOUILLOT, 2013; JUNIOR, 2012; ZHANG et al., 2012a; ABRAHAM; LAL, 2012; SKOPAL; BUSTOS, 2011; CHA, 2007; BERNDT; CLIFFORD,

1994; XUECHENG, 1992), sua relação com as técnicas de descoberta de conhecimento em banco de dados (GOLDSCHMIDT; PASSOS, 2005) e as técnicas que vêm sendo desenvolvidas para análise de similaridade semântica (GHAOUI, 2012; GAN et al., 2011; LIU; SHAO, 2010; THONGKRAU; LALITROJWONG, 2010; ZHAO; WANG; DENG, 2009; SELVAKUMAR; SUDHA, 2009; SHARIATMADARI et al., 2009; THIAGARAJAN; MANJUNATH; STUMPTNER, 2008; GABRILOVICH; MARKOVITCH, 2007).

3 TRABALHOS RELACIONADOS

Neste capítulo são analisados e comparados os trabalhos mais relevantes encontrados nas áreas de pesquisa relativas a esta dissertação. São descritos os critérios e a metodologia para a seleção dos trabalhos e apontadas as contribuições em relação aos trabalhos analisados. O texto deste capítulo está dividido em 9 seções. A primeira seção apresenta um panorama das pesquisas relacionadas, bem como a metodologia adotada para a escolha dos trabalhos. As seções de 2 a 8 descrevem os trabalhos selecionados. Por fim, na seção 9 é realizada uma análise comparativa entre os trabalhos.

3.1 Metodologia para a escolha dos trabalhos

A pesquisa foi iniciada tendo-se em mente a definição de DEY; ABOWD; SALBER (2001), de que contexto é "qualquer informação que possa ser usada para caracterizar a situação de entidades relevantes para a interação entre um usuário e uma aplicação, e que em geral é composto por dados que descrevem a situação, identidade e localização espaço-temporal de pessoas, grupos e objetos físicos ou computacionais". A partir desta definição, foram pesquisados artigos que abordassem o problema da análise de similaridade em sequências de dados, e que estes dados pudessem ser classificados como um tipo de contexto.

Ainda baseando-se na proposta de DEY; ABOWD; SALBER (2001) para categorização dos tipos de dados que podem compor o contexto, percebe-se que a categoria Tempo já está implícita no conceito de sequência ordenada cronologicamente, e a categoria Identidade implica em comparação de sequências pertencentes a entidades distintas. Seguindo esta linha de raciocínio, conclui-se que para um artigo ser considerado um trabalho relacionado, ele deve abordar a comparação de similaridade em sequências de Localizações ou de Situações/Atividades de entidades distintas.

Para trabalhos relacionados a similaridade em sequências de localizações, as pesquisas na área de similaridade de trajetórias (*Trajectory Similarity*) foram as que retornaram o maior número de possíveis trabalhos relacionados. Outra área de pesquisa relevante é a de similaridade de hábitos ou atividades de usuários, que também envolve a análise de trajetórias típicas percorridas por usuários, porém avaliando o significado dos locais visitados por cada usuário (como CASA, TRABALHO ou ESCOLA) ao invés dos padrões de deslocamento físico.

Por outro lado, a análise de similaridade de situações apresentou dificuldade maior para a localização de trabalhos relevantes. Isto deve-se principalmente ao fato de que o conceito de SITUAÇÃO possui diversos significados, conforme o domínio estudado. Para a análise de similaridade, o conceito de situação pode ser entendido como o conjunto de variáveis que descrevem uma determinada entidade em um momento específico. O tipo de dado e o significado de cada variável podem variar conforme a entidade. Podem, por exemplo, descrever condições externas como a temperatura ambiente, a luminosidade e umidade do ar, ou internas, como sinais vitais

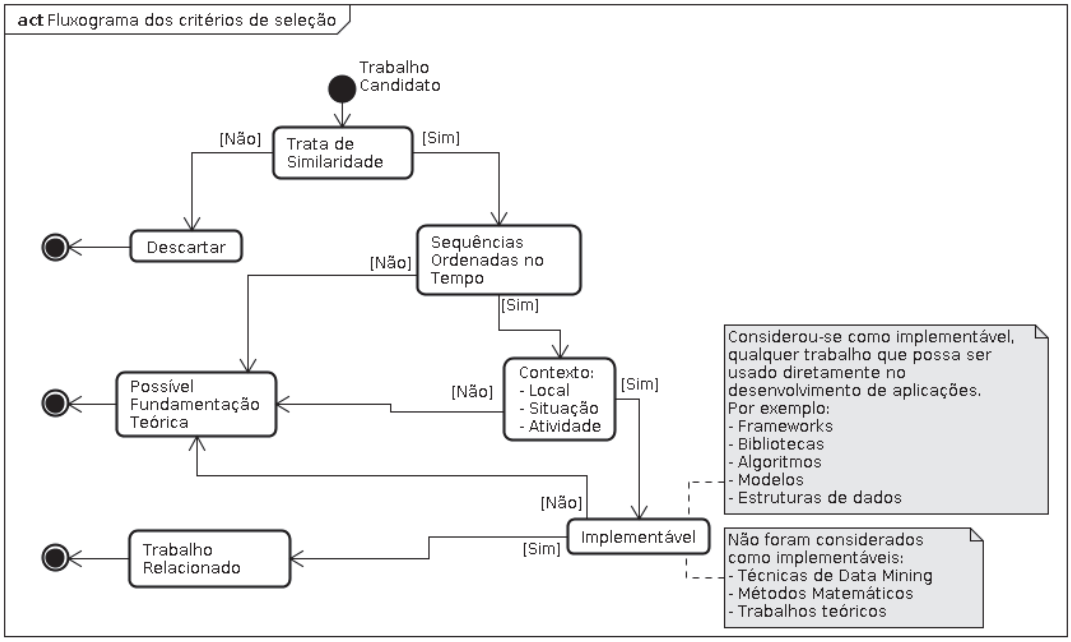
e temperatura interna. Podem ser obtidas através de sensores (como temperatura, rotação, velocidade), ou obtidas através de consultas em bancos de dados (como quantidade de clientes, produtos vendidos ou quantidade de atendentes no caixa).

Conforme demonstrado na seção 2.6, há casos em que as sequências de contextos poderão ser representadas como Séries Temporais (FU, 2011; ESLING; AGON, 2012; SOUZA, 1989), o que permite aplicar as técnicas desenvolvidas pelos pesquisadores desta área à análise de similaridade destes contextos. Por este motivo procurou-se incluir nos trabalhos relacionados, artigos que tratem da implementação de análise de similaridade entre séries temporais.

Para variáveis de situação que armazenem dados que descrevam categorias, como por exemplo o tipo de atividade, existem duas possibilidades para a análise de similaridade. A primeira seria comparar diretamente o nome das categorias em cada momento. A segunda seria utilizar uma ontologia que descreva as relações entre as categorias, o que permitiria avaliar o grau de semelhança entre duas categorias distintas. Nas pesquisas realizadas não foram encontrados trabalhos que fizessem uso de ontologias para avaliar a similaridade semântica entre duas sequências de contextos.

O processo completo de aplicação dos critérios de seleção dos trabalhos relacionados está ilustrado na Figura 24. Inicialmente foram selecionados os trabalhos das linhas de pesquisa discutidas nos parágrafos anteriores que aparentavam possuir relação com a proposta. A lista completa dos trabalhos analisados está descrita no Apêndice A. Trabalhos cujo foco não era a análise de similaridade foram descartados, e trabalhos que abordavam similaridade de forma geral e não especificamente sobre similaridade em sequências de contextos foram considerados como possíveis referências para a fundamentação teórica.

Figura 24: Critérios de Seleção de Trabalhos Relacionados



Fonte: Elaborado pelo autor

Tendo em vista a natureza prática deste trabalho estabeleceu-se que, como último critério de seleção, os trabalhos relacionados deveriam apresentar um modelo que fosse diretamente implementável, a fim de permitir a comparação com o *framework* desenvolvido. Isto inclui, mas não se restringe a, algoritmos, estruturas de dados, *frameworks*, sistemas completos e propostas de arquiteturas. Foram descartados como trabalhos relacionados aqueles de natureza teórica, mais focados em métodos matemáticos ou metodologias para *data mining*, bem como trabalhos focados no desenvolvimento de uma métrica de similaridade específica. No entanto estes ainda foram mantidos como possíveis referências à fundamentação teórica. As seções seguintes resumem os trabalhos relacionados que atenderam a estes critérios.

3.2 *Crowdsourced Trace Similarity with Smartphones*

ZEINALIPOUR-YAZTI et al. (2013) propõem um *framework* denominado *SmartTrace*$^+$ para comparar uma trajetória Q com as trajetórias geradas e armazenadas por uma multidão de *smartphones*, retornando o conjunto de trajetórias mais similares. A arquitetura do *framework* utiliza uma abordagem híbrida, composta por um servidor implementado em JAVA, responsável por gerenciar os *smartphones* conectados ao sistema e distribuir as requisições de consulta a todos os *smartphones* clientes, para que analisem localmente e de forma oportunistica, a similaridade entre a trajetória consultada e a sua própria trajetória armazenada. No final, somente os *smartphones* que reconheceram um certo grau de similaridade respondem ao servidor. Este mecanismo tem como objetivo garantir a privacidade dos usuários, visto que as trajetórias não ficam armazenadas no servidor, mas sim distribuídas entre os *smartphones* conectados, que por sua vez conhecem somente a sua própria trajetória.

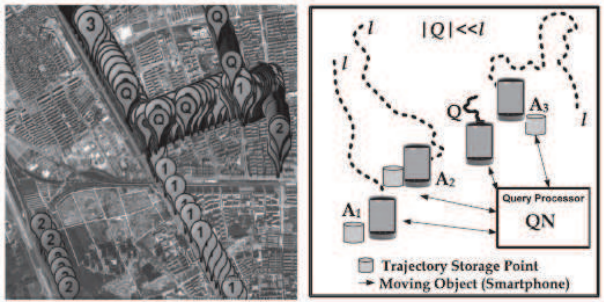
A Figura 25 ilustra a funcionalidade do *framework*. O trabalho dos autores é motivado pelo fato de que hoje em dia os *smartphones* podem armazenar as trajetórias percorridas pelos usuários durante um longo período de tempo. O objetivo é permitir a consulta *on-line* eficiente a estas trajetórias, preservando a privacidade dos usuários. A figura mostra à esquerda um conjunto de trajetórias obtidas pelo projeto *GeoLife*[1], e à direita, o modelo do sistema. O *smartphone* Q solicita ao servidor que localize trajetórias similares, que por sua vez repassa esta requisição aos demais *smartphones* conectados.

O aplicativo cliente para *smartphones* com sistema operacional *Android*™ é responsável por coletar e armazenar a trilha de locais visitados pelo usuário e responder às requisições de consulta do servidor. Também possui a funcionalidade de desenhar na tela a trajetória armazenada, conforme ilustrado na Figura 26. O aplicativo pode calcular a posição atual a partir de dois modos de operação. *Outdoor*, onde é utilizado o sinal GPS e *Indoor*, onde a posição é inferida a partir da intensidade do sinal WiFi recebido de *Access Points (AP)* próximos.

Um aspecto relevante deste trabalho é que, similar ao modelo proposto nesta dissertação, o *SmartTrace*$^+$ permite que cada *smartphone* implemente sua própria métrica de similaridade,

[1] http://research.microsoft.com/en-us/projects/geolife/

Figura 25: *SmartTrace*$^+$ *Framework*



Fonte: ZEINALIPOUR-YAZTI et al. (2013)

Figura 26: Interface gráfica para o aplicativo cliente do *SmartTrace*$^+$ *Framework*

Fonte: ZEINALIPOUR-YAZTI et al. (2013)

levando em conta que o algoritmo implementado não deve sobrecarregar o processador do *smartphone* e deve considerar questões como a economia de energia. Os autores propõem dois algoritmos a serem usados no aplicativo cliente, denominados *SmartTrace Algorithm* e *Non-Iterative SmartTrace Algorithm*. Ambos são baseados na técnica de análise de similaridade de sequências conhecida como *LCSS* (*Longest Common Subsequence*).

Um protótipo do aplicativo cliente foi implementado sobre a plataforma Android e seu pacote de instalação (APK). O servidor foi escrito em JAVA e executa sobre JDK 6 em um sistema operacional Ubuntu Linux. No futuro os autores pretendem portar as funções com maior consumo de *I/O* para código C nativo, utilizando a API Android NDK.

3.3 *Effective online group discovery in trajectory databases*

LI et al. (2012) propõem um *framework* para identificação *on-line* de grupos (*clusters*) de objetos móveis que estejam viajando juntos. Estes autores consideram a trajetória real de um objeto móvel como uma função contínua do domínio temporal $\mathbb{T}$ para pontos no espaço euclidiano n-dimensional $\mathbb{R}^n$, sendo que para a maioria das aplicações $n = 2$. Partindo desta definição, a trajetória de um objeto pode ser coletada através da amostragem de suas posições ao longo do tempo, de acordo com alguma política, resultando em um conjunto de pontos $(t, \overrightarrow{r}) \in \mathbb{T}\, X\, \mathbb{R}^n$.

Ainda segundo LI et al. (2012) existem diferentes definições para o conceito de Objetos

Viajando Juntos:

- *Flock* (Rebanho): Considera os objetos dentro de um círculo, a partir de um raio definido pelo usuário;
- *Convoy* (Comboio): Considera os objetos densamente conectados;
- *Swarm* (Enxame): Também considera a densidade das conexões, porém não exige que as sub-trajetórias sejam compostas por pontos consecutivos.

Os autores propõem então sua própria definição, que denominam Grupo. Um grupo é um *cluster* que possui pelo menos m objetos móveis conectados por densidade durante pelo menos um intervalo de tempo especificado. A Tabela 4 compara as características de cada definição.

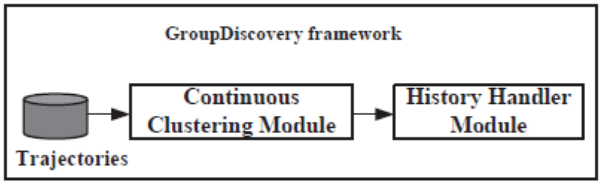
Tabela 4: Características das definições de grupos de objetos móveis

Property	***Flock***	***Convoy***	***Swarm***	***Group***
Sampling independence	✓	×	×	✓
Density connectedness	×	✓	✓	✓
Traj. approximation	✓	✓	✓	✓
Online processing	✓	×	×	✓

Fonte: LI et al. (2012)

A arquitetura do *framework* está ilustrada na Figura 27. O componente *Continuous Clustering Module* inicialmente utiliza a técnica conhecida como *DBScan* (*Density-Based Clustering*) para identificar grupos com uma extensão espacial arbitrária. Estes grupos são então monitorados, bem como a chegada de novos dados. A entrada ou saída de novos objetos em um *cluster*, bem como a expiração e combinação de *clusters* são modelados como eventos futuros e processados quando realmente ocorrerem. Esta abordagem orientada a eventos garante a independência do método de amostragem, bem como o processamento *on-line* dispensando a necessidade de recalcular todos os *clusters* sempre que um novo dado é recebido. O componente *History Handler Module* é responsável por gerenciar de forma eficiente o histórico de cada grupo, para encontrar grupos a serem retornados ao usuário.

Figura 27: Arquitetura do *GroupDiscovery Framework*



Fonte: LI et al. (2012)

Os eventos são modelados no *framework* como uma tupla $(t, obj, cid_1, cid_2, type)$, onde t é o horário do evento, obj é o identificador único do objeto, cid_1 e cid_2 são identificadores de *clusters*, e *type* identifica o tipo de evento.

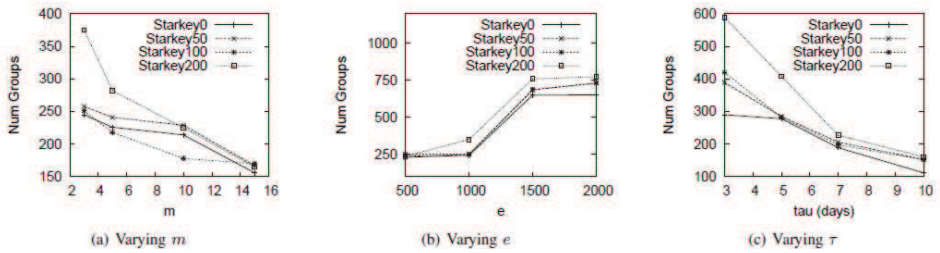
- *APPEAR*: Um novo objeto é recebido pelo sistema;
- *DISAPPEAR*: Um objeto deixa de existir, para o sistema;
- *UPDATE*: Um objeto altera sua velocidade, o que afeta o tempo previsto para sua saída do *cluster*;
- *EXIT*: Um objeto deixa de fazer parte de um *cluster*;
- *JOIN*: Um objeto se junta a um *cluster*;
- *EXPIRE*: O tempo de duração de um *cluster* expira, o que faz com que seus objetos passem a constar como não-classificados;
- *MERGE*: Uma união de dois ou mais *clusters*;
- *SPLIT*: Uma divisão de um *cluster* em dois ou mais *clusters*.

Os objetos gerenciados pelo *framework* são representados por uma tupla $(oid, \overrightarrow{p}, \overrightarrow{v}, cid, label)$, onde oid é o identificador único do objeto, $\overrightarrow{p}$ é a sua localização, $\overrightarrow{v}$ é a sua velocidade, cid é o identificador do *cluster* ao qual o objeto está vinculado e $label$ é o tipo de objeto. A técnica *DBScan* considera dois tipos de objetos. Os *Core Objects* são aqueles que contêm pelo menos *MinPts* em sua vizinhança, onde *MinPts* é um valor definido pelo usuário. No trabalho de LI et al. (2012) é utilizada a varíavel m da definição de Grupo. Os demais objetos são chamados de *Border Objects*.

Um protótipo deste *framework* foi implementado na linguagem JAVA e testado em um computador com processador Intel™ Xeon Quad-Core(2.66Ghz), 8 GB de memória RAM e sistema operacional Linux (Kernel 2.6.32). Foram utilizadas três bases de dados de trajetórias, obtidas de veículos e animais, e uma base sintética gerada por um *software* gerador de trajetórias chamado GSTD (THEODORIDIS; SILVA; NASCIMENTO, 1999).

Foram realizados experimentos variando o valor dos parâmetros m (cardinalidade esperada para os grupos), e (limite de distância) e τ (tempo de vida). A Figura 28 ilustra os efeitos da variação destes parâmetros sobre a quantidade de grupos encontrados em amostras do mesmo *dataset*.

Figura 28: Efeitos da variação de m, e e τ nos grupos identificados



Fonte: LI et al. (2012)

3.4 *Spatio-temporal similarity of network contrained moving object trajectories using sequence alignment of travel locations*

ABRAHAM; LAL (2012) e ABRAHAM; LAL (2011) propõem dois *frameworks* para análise de similaridade em sequências de trajetórias. O primeiro, denominado *TraSimilar* tem como objetivo a formação de *clusters* de veículos com trajetórias similares, considerando o histórico de POI (*Point-Of-Interest*) e TOI (*Time-Of-Interest*) de cada veículo. O segundo, denominado *WebTraSim* aplica o conceito de trajetória à sequência de páginas *web* visitadas por um usuário, criando assim uma métrica de similaridade de trajetórias que considera o histórico de UOI (*URL-Of-Interest*) e TOI (*Time-Of-Interest*) de cada usuário. Os autores argumentam que a análise da sequência de páginas visitadas, obtidas a partir dos *logs* de acessos da *dark web*, pode ajudar a descobrir padrões de acesso a páginas terroristas.

A técnica implementada em ambos os *frameworks* é a mesma, e permite avaliar a similaridade espacial, temporal, e espaço-temporal de duas trajetórias. A análise espacial considera três fatores:

1. conjunto de localizações comuns visitadas por ambas as entidades;
2. similaridade estrutural das localizações nas trajetórias;
3. similaridade da sequência de ambas as trajetórias.

O conjunto de localizações comuns é dado pela intersecção dos conjuntos de localizações de ambas as entidades, desconsiderando-se a sequência de visitação, em relação ao conjunto total de localizações. Considerando-se duas trajetórias T_i e T_j, a similaridade do conjunto de localizações é dada por:

$$Sim_c(T_i, T_j) = \frac{Quantidade\ de\ locais\ visitados\ tanto\ por\ T_i\ quanto\ por\ T_j}{Total\ de\ locais\ visitados\ por\ T_i\ e\ T_j} \Rightarrow \frac{T_i \cap T_j}{T_i \cup T_j}$$

Os autores utilizam o termo Similaridade Estrutural para se referir a forma de comparação de cada par de itens de duas trajetórias. A técnica para avaliar a similaridade estrutural varia conforme o domínio da aplicação. Em ABRAHAM; LAL (2012) a similaridade mede a proximidade geográfica entre dois pontos das trajetórias. Já em ABRAHAM; LAL (2011) a similaridade estrutural é dada pelo caminho completo de cada URL, dentro do servidor. As técnicas para análise da similaridade utilizadas nestes trabalhos é descrita a seguir.

- **Similaridade estrutural para trajetórias de veículos** (ABRAHAM; LAL, 2012): Cada localização é codificada em uma *String* binária composta por *Country* (dois *bits*), *District* (quatro *bits*), *Road* (quatro *bits*) e *Relative road location* (quatro *bits*). Por exemplo: 00 1010 0011 0010, representa o país 00, o distrito 1010, a estrada 0011 e a localização relativa na estrada 0010.

A similaridade entre duas *strings* é obtida considerando-se cada *substring* de *bits* separadamente. É atribuído um peso crescente para cada *bit* da esquerda para a direita, na ordem $1, 2, 3, .., n$ onde n é o tamanho da *string*. A similaridade é então medida somando-se os pesos dos *bits* que são iguais em ambas as sequências. Considerando-se dois pontos P_i e P_j, representados por *Strings* binárias compostas por *Country* $_c$, *District* $_d$, *Road* $_r$ e *Relative road location* $_l$, a similaridade de cada *substring* é dada por:

$Sim_s(P_i, P_j) = \frac{Soma\ dos\ pesos\ dos\ bits\ iguais\ nas\ duas\ Strings}{Soma\ dos\ pesos\ das\ Strings}$

Exemplo: Considerando-se as *substrings* $P_i^d = 0001$ e $P_j^d = 0010$, e a sequência de pesos $(1, 2, 3, 4)$, a similaridade estrutural é dada por $Sim_{sd}(P_i, P_j) = \frac{1+2}{1+2+3+4} = 0.3$

O *framework* somente avalia a *substring* seguinte se encontrar uma similaridade total $(Sim_s = 1)$ na *substring* atual. Por exemplo, se o país de P_i for diferente do país de P_j, o algoritmo encerra e considera que $Sim_{sd}(P_i, P_j) = Sim_{sr}(P_i, P_j) = Sim_{sl}(P_i, P_j) = 0$.

Por fim, a similaridade total entre as *Strings* é dada por:

$Sim_s = \frac{Sim_{sc}(P_i,P_j)+Sim_{sd}(P_i,P_j)+Sim_{sr}(P_i,P_j)+Sim_{sl}(P_i,P_j)}{4}$

- **Similaridade estrutural para históricos de acessos a páginas *web*** (ABRAHAM; LAL, 2011): Cada ponto da trajetória é composto pelo caminho de acesso a determinada página, dentro da estrutura de diretórios do servidor, por exemplo: $P1 =$"/faculty/pub/journal/ieee.htm" e $P2 =$"/faculty/pub/conference.htm". O conteúdo de cada nível é representado por *token* e a concactenação de *tokens* forma a *string* representativa da *URL*. Assim como nas trajetórias de veículos (ABRAHAM; LAL, 2012), a medida da similaridade é dada atribuindo-se um peso para cada *token*, somando-se os pesos dos *tokens* iguais e dividindo-se pela soma de todos os pesos. A diferença neste caso é que o tamanho das *strings* pode ser diferente. Por este motivo os autores introduziram variável L_{longer} que corresponde ao tamanho (*Length*) da maior *string*.

  Considerando-se:

  $P1 =$"/faculty/pub/journal/ieee.htm" e $P2 =$"/faculty/pub/conference.htm".

  - $L_{longer} = 4 \Rightarrow$ (Quantidade de tokens em $P1$);
  - *Token String* $P1$: 0 2 2 2;
  - *Token String* $P2$: 0 2 1;
  - Peso de cada *token:* 4, 3, 2, 1 $\Rightarrow$ (A sequência de pesos é inversa a sequência de pesos da análise de veículos).

  A similaridade estrutural entre estas páginas é dada por:

  $Sim_s(P1, P2) = \frac{4+3}{4+3+2+1} = 0.7$

Tanto a medida de similaridade de conjuntos quanto a similaridade estrutural satisfazem as seguintes propriedades:

1. $Sim(T_i, T_j) = Sim(T_j, T_t)$;
2. $0 \leq Sim(T_i, T_j) \leq 1$;
3. $Sim(T_i, T_j) = 0$, quando não houver nenhuma similaridade entre as trajetórias;
4. $Sim(T_i, T_j) = 1$, quando as trajetórias forem exatamente as mesmas.

Para avaliar a Similaridade das Sequências das trajetórias, tanto de veículos (ABRAHAM; LAL, 2012) quando de páginas *web* (ABRAHAM; LAL, 2011), o *framework* aplica o seguinte procedimento:

1. **Alinhamento das trajetórias, baseado nas localizações**.

   Considerando-se $T_1 = (P_1, P_5, P_7, P_3, P_6)$ e $T_2 = (P_1, P_5, P_3, P_6)$, são inseridos *gaps* nas sequências, de forma a obter o melhor alinhamento possível.

   $T_1 = P_1, P_5, P_7, P_3, P_6$

   $T_2 = P_1, P_5, \_\_, P_3, P_6$

2. **Criação da matriz de similaridade**.

   Uma das sequências é colocada no topo da matriz, e a outra sequência é colocada a esquerda. O valor 0 (zero) é armazenado na célula superior esquerda e corresponde ao ponto de início do cálculo de similaridade.

   A partir da Similaridade Estrutural entre cada par de pontos, é obtido um *score* de similaridade, que pode variar de -10 até +20. Este *score* é calculado a partir da fórmula $score = -10 + 30 * Sim_s(P_i, P_j)$. O valor da célula é então preenchido com o maior valor da soma do *score* obtido com o valor da célula a esquerda, superior, ou diagonal superior esquerda, dado pela fórmula:
   $valor[i, j] = max(score{+}valor[i{-}1, j], score{+}valor[i, j{-}1], score{+}valor[i{-}1, j{-}1])$

	__	P1	P5	P7	P3	P6
__	0	-10	-20	-30	-40	-50
P1	-10	20	10	0	-10	-20
P5	-20	10	40	30	20	10
P3	-30	0	30	30	50	40
P6	-40	-10	20	20	40	70

3. **Cálculo da similaridade entre as sequências**: O melhor valor de *score* da matriz (OSSV - *Optimal Similarity Score value*) é lido a partir da célula no canto inferior direito. O valor da similaridade entre as sequências é então dado pela fórmula:

   $Sim_q(T_i, T_j) = \frac{OSSV}{20 * L_{shorter}}$

   Onde $L_{shorter}$ é o tamanho (*Length*) da menor trajetória.

Por fim o *framework* calcula a similaridade global entre as trajetórias através da média entre a Similaridade de Conjuntos e a Similaridade de Sequências, através da fórmula:

$Ssim_{cs}(T_i, T_j) = \frac{Sim_c(T_i,T_j)+Sim_q(T_i,T_j)}{2}$

Após calcular a similaridade espacial, o *framework* executa um refinamento, baseado na similaridade temporal entre os locais visitados, sejam páginas ou veículos. Inicialmente, é calculada a distância temporal dos pontos em comum entre as duas sequências, que é dada pela diferença de tempo entre o horário de visitação do mesmo ponto, pelas duas trajetórias. O somatório destas diferenças corresponde a distância temporal, e é dada pela função:

$dist_T(T_i, T_j) = \frac{1}{m * t_s} \sum |T_i.t - T_j.t|$

Onde m é a quantidade de locais em comum entre as duas trajetórias e t_s é o intervalo de tempo total correspondente as duas trajetórias. Esta função apresenta as seguintes propriedades:

1. $0 \leq dist_T(T_i, T_j) \leq 1$;
2. $dist_T(T_i, T_j) = dist_T(T_j, T_i)$;

Apenas serão considerados similares os pontos em que a distância de tempo for menor ou igual ao valor máximo, definido pelo parâmetro σ, configurado pelo usuário.

O *WebTraSim* (ABRAHAM; LAL, 2011) foi implementado em *Visual Basic*™ e testado em um computador com processador *Core-2 Duo* 2.8GHz, 1GB de memória RAM e Sistema Operacional *Windows XP Professional*™. A eficiência do *framework* foi testada em um conjunto de dados contendo os *logs* de acesso a um servidor *web*, no dia 26 de agosto de 2009, com 47.685 requisições e 1.986 sessões de usuários. Os autores concluíram que o desempenho deste algoritmo é igual a de outros algoritmos similares.

O *TraSimilar* (ABRAHAM; LAL, 2012) foi testado em um computador com processador *Core-2 Duo* 2.8GHz, 2GB de memória RAM e Sistema Operacional *Windows XP Professional*™. Os autores não especificaram em qual linguagem o protótipo foi implementado. A eficiência do *framework* foi testada em conjuntos de dados de trajetórias de veículos e os autores concluíram que o tempo médio de resposta do *framework* aumenta linearmente em função do número de POI's.

3.5 *Mining user similarity based on routine activities*

Segundo LV; CHEN; CHEN (2013) a análise de similaridade de sequências de contextos é um fator crucial para a nova geração de redes sociais baseadas em localização (*LBSN - Location Based Social Network*), e que portanto vem atraindo muita atenção dos pesquisadores. Porém muitos dos trabalhos existentes se focam em analisar as sequências de posições geográficas ou de características de usuários. A limitação desta abordagem é que as trajetórias físicas em geral correspondem a períodos curtos de tempo, impossibilitando uma análise de similaridade a longo prazo.

Para resolver este problema, LV; CHEN; CHEN (2013) propõem uma abordagem em dois estágios. No primeiro, a noção de Atividades de Rotina é proposta para capturar a regularidade de atividades de um usuário a longo prazo. A sequência de rotinas é inferida a partir dos dados de suas trajetórias diárias. No segundo estágio, a similaridade é calculada de forma hierárquica, baseada nas atividades de rotina extraídas.

A Figura 29 ilustra esta abordagem. Inicialmente calcula-se a matriz de locais de preferência do usuário, denominada *1-Day Activity* (**a**). O valor em cada célula corresponde a quantidade de minutos que o usuário esteve no local durante o intervalo de tempo. A *Routine Activity* é uma matriz de distribuição de probabilidades (**b**), onde o valor da célula corresponde a probabilidade do usuário estar neste local, neste período de tempo.

Figura 29: Representação das matrizes *1-Day activity* e *Routine Activity*

(a)

	8:00~9:00	9:00~10:00	...	19:00~20:00	20:00~21:00
home	21	0	...	5	60
lab	7	60	...	25	0
on the way	32	0	...	30	0

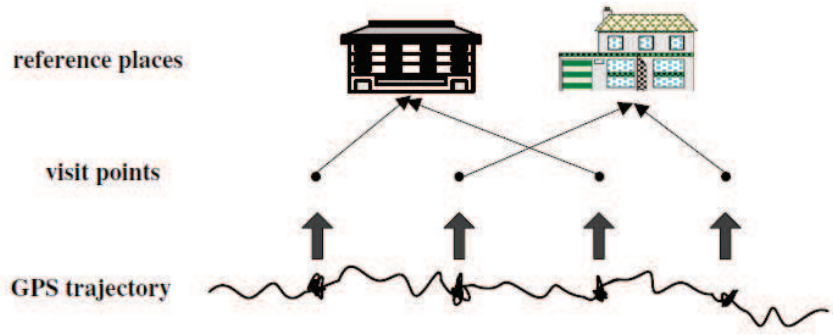
(b)

	8:00~9:00	9:00~10:00	...	19:00~20:00	20:00~21:00
home	0.4	0	...	0.1	0.95
lab	0.1	0.98	...	0.35	0
...	...	...	...	...	...
on the way	0.5	0.02	...	0.55	0.05

Fonte: LV; CHEN; CHEN (2013)

A identificação dos locais visitados pelos usuários é chamada pelos autores de Locais de Referência e inferida a partir dos Pontos de Visitação, que por sua vez são obtidos a partir da trajetória do GPS. Estes conceitos estão ilustrados na Figura 30.

Figura 30: Extração de locais de referência baseada em clusterização hierárquica

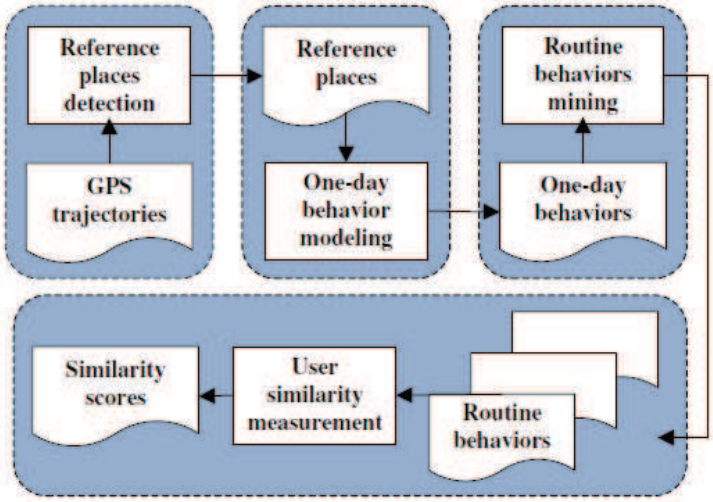


Fonte: LV; CHEN; CHEN (2013)

A Figura 31 apresenta uma visão geral da arquitetura do sistema. Dadas as trajetórias de

GPS de múltiplos usuários, a abordagem de LV; CHEN; CHEN (2013) calcula o *score* de similaridade de cada par de usuários em quatro passos. Para cada usuário individual, o sistema primeiro extrai os locais de referência a partir das trajetórias. Em seguida o sistema transforma as trajetórias obtidas do GPS em uma matriz *1-Day Activity*. O sistema então extrai os padrões desta matriz para compor a matriz de Atividades de Rotina. Por fim, o sistema mede a similaridade dos usuários baseando-se nas suas respectivas matrizes de Atividades de Rotina.

Figura 31: Arquitetura do sistema de mineração de atividades de usuários



Fonte: LV; CHEN; CHEN (2013)

Dadas duas matrizes de Atividades de Rotina, elas são similares quando possuírem alta probabilidade em comum de visitar Locais de Referência nos mesmos intervalos de tempo.

3.6 *A Framework for Similarity Search of Time Series Cliques with Natural Relations*

Segundo CUI; ZHAO; TOK (2012) *Time Series Cliques* consistem em múltiplas séries temporais naturalmente relacionadas umas as outras. Por exemplo, em uma partida de futebol os movimentos de cada jogador podem ser descritos por uma série temporal. O conjunto de séries temporais que representam os movimentos de todos os jogadores da mesma partida é chamado de *Time Serie Clique* (*TSC*).

Os autores desenvolveram um *framework* para busca eficiente de similaridade em dados de TSC's. Primeiramente, o *framework* oferece uma forma compacta de representar os dados de uma TSC. Em seguida é utilizado um vetor multidimensional para mapear as relações entre as várias séries temporais em uma TSC. Finalmente, o *framework* define uma nova medida de similaridade utilizando a representação compacta e o vetor multidimensional proposto.

Formalmente, CUI; ZHAO; TOK (2012) definem a TSC como um conjunto $\{T_1, T_2, T_3, ..., T_n\}$, onde $T_i = \{(x_m, t_m)||l = 1, 2, ..., L_i\}$ refere-se a uma série temporal individual e n é o número de séries na TSC. Cada $T_i, x_m \in \mathbb{R}^k$ representa um valor observado em um momento específico,

onde k representa a dimensionalidade de cada elemento (x_m) na série temporal (T_i). $t_m \in \mathbb{R}^1$ representa um momento específico e L_i é o tamanho da i-nésima série temporal.

O problema da busca de similaridade em TSC's consistem em encontrar a TSC T_s mais similar a TSC Q em um *DataBase* $D = \{T_1, T_2, ..., T_n\}$, através de uma função pré-definida $dist(Q, T_i)$, onde s satisfaz a condição $s = min(dist(Q, T_i))$, $i = 1, 2, ..., n$. Da mesma forma que o *SmartTrace*$^+$ (ZEINALIPOUR-YAZTI et al., 2013), o modelo proposto por CUI; ZHAO; TOK (2012) é independente da métrica de similaridade escolhida, podendo ser implementado com qualquer função de distância desejada.

O modelo proposto pelos autores é consiste em duas estruturas de dados que unificam a relação das séries temporais com seus formatos ou padrões. A primeira estrutura é chamada de *Relation-Vector for TSC* (RT) que descreve as relações naturais que existem entre as várias séries temporais que compõem a TSC. A segunda estrutura é chamada de *Compact Representation for TSC* (CT) e é construída aplicando-se a técnica de Análise de Componentes Principais (*PCA*) para reduzir a dimensionalidade das séries temporais.

Após a geração das estruturas *RT* e *CT* para cada TSC, a busca de similaridade proposta por CUI; ZHAO; TOK (2012) consiste em, dadas duas TSC's representadas por suas respectivas estruturas, $TSC_i = (CT_i, RT_i)$ e $TSC_j = (CT_j, RT_j)$, aplicar uma função de distância qualquer (por exemplo, a distância euclidiana), sobre estas estruturas ($dist_C(CT_i, CT_j)$ e $(dist_R(RT_i, RT_j))$ e combinar os resultados para obter a similaridade global entre as TSC's. Por fim, os autores argumentam que uma aplicação pode, além de escolher a métrica de distância, atribuir pesos diferentes para $dist_C$ e $dist_R$ conforme sua necessidade.

O *framework* proposto pelos autores foi implementado utilizando o *MatLab* e os testes foram realizados em um computador com CPU *Core* 8 2.8 GHZ CPU e Sistema Operacional Ubuntu Linux. Segundo os autores, a análise de desempenho realizada a partir dos testes mostrou claramente a superioridade desta abordagem para busca de similaridade em bancos de dados de séries temporais.

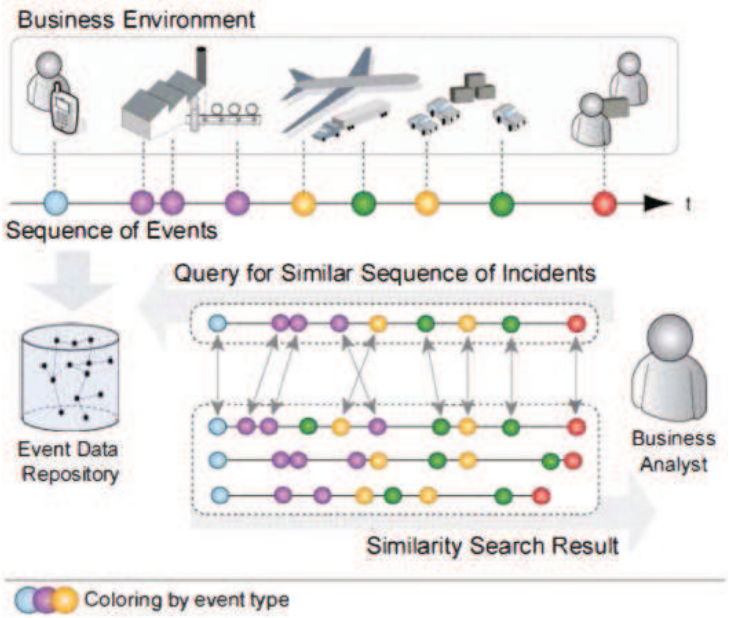
3.7 *Similarity searching in sequences of complex events*

OBWEGER et al. (2010) propõem um modelo para análise de similaridade em sequências de eventos. Segundo estes autores, o método para Processamento de Eventos Complexos conhecido como *CEP* (LUCKHAM, 2001), permite às empresas monitorar em tempo real incidentes relativos a suas operações e implementar decisões automáticas para reagir a possíveis ameaças em seus negócios. O modelo proposto pelos autores busca responder a seguinte questão: Tendo em mãos uma sequência de eventos, quais outras sequências históricas são semelhantes?

Para ilustrar a aplicação deste modelo, OBWEGER et al. (2010) utilizam como exemplo um cenário em uma empresa de apostas *on-line*, onde após detectar um novo método de fraude, analistas podem buscar nos dados históricos indícios de transações antigas onde este método foi aplicado.

O modelo proposto pelos autores baseia-se na premissa de que, intencionalmente ou não, as pessoas derivam a similaridade percebida a partir de um padrão virtualmente estabelecido S_p, o qual é mapeado da melhor maneira possível em uma sequência candidata S_c. Isto significa que cada elemento em S_p possui pelo menos um equivalente conceitual distinto em S_c. O valor final da similaridade resulta a partir dos desvios entre o padrão de consulta S_p e o seu mapeamento em S_c. A Figura 32 ilustra este processo, iniciando a partir de um conjunto de eventos reais e seus mapeamentos em sequências históricas de eventos.

Figura 32: Busca de similaridade em sequências de eventos



Fonte: OBWEGER et al. (2010)

Para OBWEGER et al. (2010) os seguintes requisitos são necessários a este tipo de aplicação:

- **Generalidade**: O modelo deve ser aplicável a diversos tipos de dados;
- **Configurabilidade**: O modelo deve ser orientado ao interesse do usuário, o que significa que é o usuário quem determina o que ele considera similar, e quais os critérios para avaliar a similaridade em sequências;
- **Suporte a modos de combinação**: O modelo deve permitir que a análise de similaridade seja aplicada considerando-se as sequências completas ou apenas parte delas.

Os autores argumentam que este é o primeiro trabalho a oferecer uma medida de similaridade compreensível para sequências de eventos, já que outros trabalhos como o de DEY; SARKAR; DE (2002) concentram-se na similaridade de eventos individuais.

3.8 Análise comparativa dos trabalhos selecionados

Nesta seção é feita uma comparação dos trabalhos relacionados em função de suas características, funcionalidades e relação com o *framework* proposto. Foram avaliados quatro grupos de características.

1. **Categorias de dados contextuais**: Avalia o domínio no qual o trabalho atua, procurando classificar a natureza dos dados analisados através das categorias de dados contextuais propostas por DEY; ABOWD; SALBER (2001);
2. **Tipos de similaridade consideradas**: Avalia como o trabalho calcula a similaridade de cada par de item das sequências. Para fins de comparação, identificou-se quatro tipos de cálculo de similaridade.
   (a) Por valor numérico: Quando os itens são compostos por valores numéricos e a similaridade entre eles pode ser dada diretamente através de uma equação. Neste grupo se enquadram valores escalares como temperatura ou umidade, vetoriais como deslocamento, e também posições geográficas definidas por latitude e longitude;
   (b) Por semântica: É avaliado o significado da descrição de cada item ou a categoria na qual cada descrição se enquadra. Por exemplo, Casa ou Universidade são categorias de localizações;
   (c) Por semântica, através de ontologias: Quando a análise semântica é realizada com o auxílio de ontologias, o que permite identificar relações entre itens, mesmo que suas descrições sejam diferentes, desde que ambas as descrições estejam mapeadas como classes de uma ontologia;
   (d) Por sintaxe: Compara a *string* que representa cada item, admitindo a comparação sintática de palavras ou *sub-strings*;
3. **Formas de análise das sequências**: Avalia como o trabalho calcula a similaridade global de duas sequências. Foram identificados três formas de análise.
   (a) Dados sumarizados: Realiza um pré-processamento para criar um resumo dos dados das duas sequências e realiza a análise baseado neste resumo;
   (b) Item-a-item: Compara cada par de item das duas sequências, aplicando ou não alguma técnica de alinhamento, e ao final calcula a similaridade global através de somatório, média ou alguma outra técnica;
   (c) Intersecção de conjuntos: Divide a quantidade de itens em comum nas duas sequências pela quantidade total de itens únicos. Pode ser formalizada como: $Sim(S_i, S_j) = \frac{S_i \cap S_j}{S_i \cup S_j}$, onde S_i e S_j são duas sequências de contextos;
4. **Recursos oferecidos**: Avalia as funcionalidades oferecidas pelo trabalho aos desenvolvedores de aplicações.

(a) Pré-Processamento: Avalia se o trabalho oferece recursos de pré-processamento sobre os dados das sequências, como por exemplo, redução de dimensionalidade ou seleção de amostras;

(b) Pós-Processamento: Avalia se o trabalho oferece recursos de pós-processamento sobre os dados analisados, como por exemplo, filtragem de resultados ou realização de inferências;

(c) Multi-Métrica: Avalia se o trabalho oferece suporte à utilização combinada de diversas métricas de similaridade;

(d) Métrica Própria: Avalia se o trabalho desenvolveu uma métrica própria para análise de similaridade ou se utilizou uma métrica já existente;

(e) Seleção de Métrica: Avalia se o trabalho permite que o desenvolvedor selecione uma das métricas de análise de similaridade oferecidas ou implementar sua própria métrica;

(f) Configurável: Avalia se o trabalho permite configurar os parâmetros de operação e cálculo de similaridade.

A Tabela 5 compara os trabalhos utilizando os critérios apresentados. Como demonstrado nesta tabela, a maioria dos trabalhos relacionados encontrados nesta pesquisa concentram-se em um domínio específico. O *SmartTrace*$^{+}$ (ZEINALIPOUR-YAZTI et al., 2013) e o *Routine Activity* (LV; CHEN; CHEN, 2013) analisam a similaridade de localizações geográficas ou descritivas de usuários. O *WebTraSim* (ABRAHAM; LAL, 2012) analisa a similaridade de sequências de páginas *web* visitadas por usuários. O *Group Discovery Framework* (LI; EICKHOFF; VRIES, 2012) e o *TraSimilar* (ABRAHAM; LAL, 2011) analisam a similaridade de objetos em trânsito. O *SeqSim* (OBWEGER et al., 2010) analisa sequências de eventos registradas em históricos de transações. Por último, o único trabalho identificado que tem como objetivo realizar análises de similaridade independente do domínio é o *TSC* (CUI; ZHAO; TOK, 2012), que analisa similaridade de dados que possam ser descritos como uma série temporal.

Não foram identificados trabalhos que analisem a similaridade semântica com o uso de ontologias. Este recurso permitiria ao *framework* oferecer uma maior flexibilidade, pois através dela seria possível configurar os relacionamentos entre as categorias e os termos existentes em um determinado domínio.

Nenhum dos trabalhos identificados oferecem recursos de pós-processamento. Procedimentos de pós-processamento permitiriam refinar os resultados encontrados por estes trabalhos, como por exemplo: ordenação de resultados, filtros, aplicação de tarefas de *KDD* (*knowledge-discovery in databases*) sobre os resultados da análise, execução de procedimentos automáticos conforme os resultados da análise, entre outras aplicações.

Por fim, todos os trabalhos relacionados encontrados aplicam apenas uma medida de similaridade, não permitindo a combinação de métricas diferentes para calcular a similaridade global entre duas sequências. A maioria dos trabalhos identificados utiliza sua própria métrica

Tabela 5: Comparativo dos trabalhos relacionados

Características		SmartTrace+ ZEINALIPOUR-YAZTI et al. (2013)	Group Discovery Framework LI et al. (2012)	TraSimilar ABRAHAM; LAL (2012)	WebTraSim ABRAHAM; LAL (2011)	Routine Activity LV; CHEN; CHEN (2013)	TSC CUI; ZHAO; TOK (2012)	SeqSim OBWEGER et al. (2010)
Categoria de dados contextuais DEY; ABOWD; SALBER (2001)	Entidade *(Identity)*	Usuários de Smartphone	Genérica	Veículos	Usuários Web	Usuários	Genérica	Business Events
	Tempo *(Hora, Data, ...)*	Independe	Data / Hora	Data / Hora	Data / Hora	Hora do dia	Independe	Data / Hora
	Posição Geográfica *(Location)*	SIM	SIM	SIM	não	SIM	SIM	não
	Situação *(Descrita por variáveis)*	não	não	não	não	não	SIM	SIM
	Atividade *(O que a entidade está fazendo)*	não	não	não	SIM	SIM	não	SIM
Tipo de Similaridade	Valor numérico	SIM	SIM	não	não	não	SIM	não
	Semântica	não	não	não	não	SIM	não	SIM
	Ontologias	não	não	não	não	não	não	não
	Sintaxe	não	não	SIM	SIM	não	não	não
Forma de análise	Dados Sumarizados	não	não	não	não	SIM	SIM	não
	Item-a-Item	SIM	SIM	SIM	SIM	não	não	SIM
	Intersecção de Conjuntos	não	não	SIM	SIM	não	não	não
Recursos oferecidos	Pré Processam.	não	não	SIM	SIM	SIM	SIM	não
	Pós Processam.	não	não	não	não	não	não	não
	Multi-Métrica	não	não	não	não	não	não	não
	Métrica própria	SIM	não	SIM	SIM	SIM	SIM	SIM
	Seleção de métrica	SIM	não	não	não	não	SIM	não
	Configurável	não	SIM	não	não	não	não	SIM

Fonte: Elaborado pelo autor

para calcular a distância entre itens e apenas o *TSC* (CUI; ZHAO; TOK, 2012) permite escolher livremente a função de distância.

3.9 Considerações sobre o capítulo

Neste capítulo foi realizado um estudo comparativo dos trabalhos relacionados. Inicialmente foram realizadas buscas em bases de dados de periódicos a procura de pesquisas que pudessem ser comparadas ao *framework* especificado nesta dissertação. Percebeu-se que os trabalhos que abordam a questão da similaridade em sequências de dados contextuais em geral concentram-se em domínios e métricas específicas.

A especificação de um *framework* extensível e configurável, que permitisse combinar diferentes métricas ofereceria um mecanismo mais apropriado à análise de similaridade em sequências de dados contextuais que, como demonstrado nas seções 1.1 e 1.2, são naturalmente hete-

rogêneos, podendo exigir técnicas específicas de análise para cada dimensão do contexto.

Por fim, a análise de similaridade semântica em sequências de contextos, auxiliada pelo uso de ontologias, é uma característica que não estava presente em nenhum dos trabalhos analisados. Tendo em vista estas questões, é proposta a especificação de um *framework* para análise combinada de similaridade em sequências de contextos de entidades genéricas, descrito no próximo capítulo.

4 ESPECIFICAÇÃO DO *SIMCOP*

Este capítulo descreve a especificação da arquitetura do *SIMCOP*. A primeira seção introduz os aspectos conceituais do *framework*. Na segunda seção é feito o levantamento de requisitos. Na terceira seção é apresentada a arquitetura. A seção quatro descreve a utilização do *SIMCOP*. Por fim, a seção cinco apresenta as considerações finais do capítulo.

4.1 Visão Geral

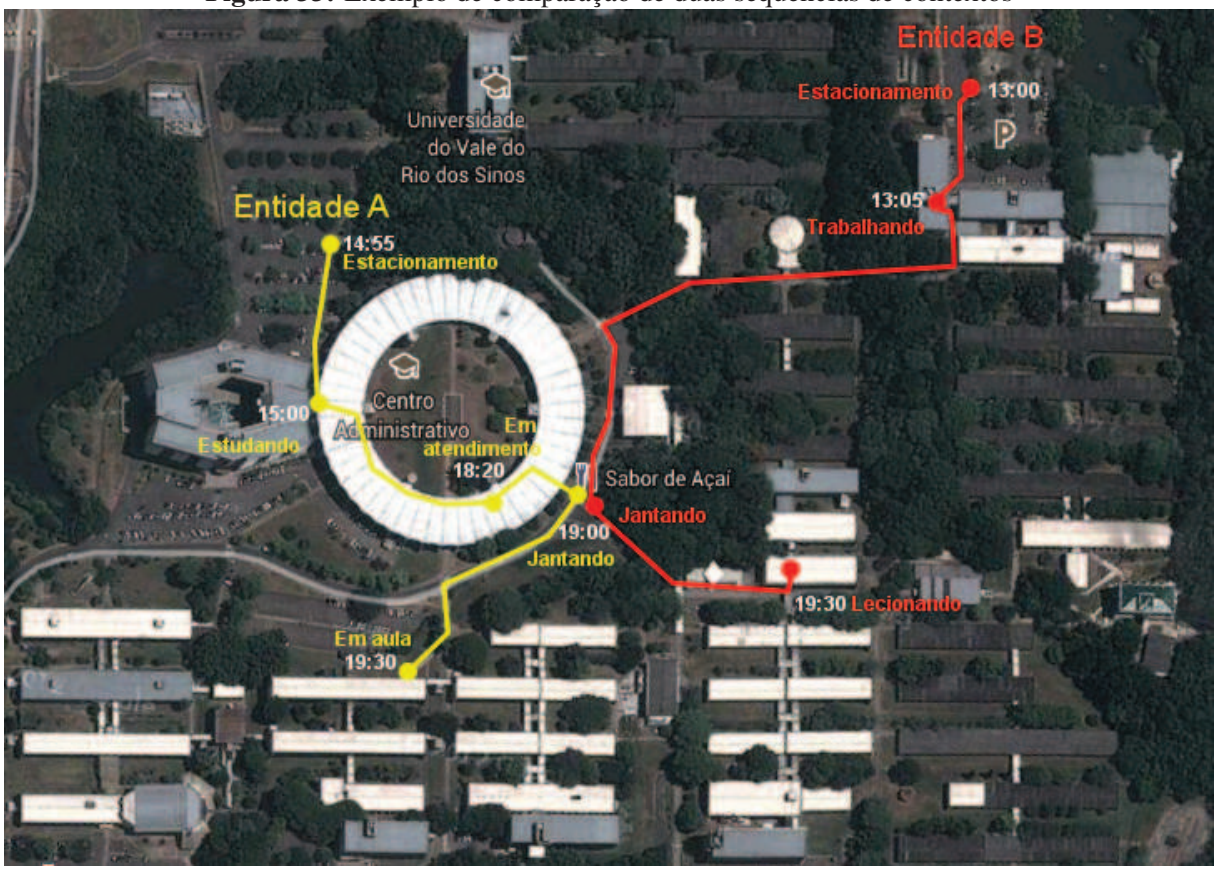
O objetivo deste *framework* é auxiliar o desenvolvimento de aplicações que necessitem analisar a similaridade entre sequências de contextos. A dificuldade para realizar este tipo de análise é devida principalmente a natureza heterogênea dos dados que compõem a descrição contextual de entidades genéricas. Isto gera a necessidade da utilização de diferentes técnicas de medição de similaridade durante uma mesma análise e, de alguma forma, combinar os resultados individuais destas técnicas para obter um resultado global da análise.

A Figura 33 ilustra um exemplo de comparação de similaridade entre as sequências de contextos de duas entidades distintas, no caso um aluno (Entidade A) e um professor (Entidade B) da Unisinos. As linhas no mapa correspondem ao percurso (*position*) percorrido por ambas as entidades dentro do *campus* da universidade. Os pontos em cada linha correspondem a *Places of Interest* (*POI*) visitados por cada entidade durante o percurso (*location*), o horário (*time*) da visitação e a atividade (*situation*) na qual a entidade estava envolvida. Neste exemplo, ocorrem três similaridades entre as sequências. A primeira é o ponto de início de cada sequência, onde apesar da posição e do horário serem diferentes, as atividades são as mesmas e são exercidas por ambas as entidades no mesmo tipo de *POI* (Estacionamento). Considerando-se este cenário, é possível inferir que ambas as entidades, ao chegarem ao campus, estacionam os seus veículos em um estacionamento. A segunda similaridade ocorre às 19:00, onde ambas as entidades estão jantando, no mesmo local, e fisicamente próximas. Por fim, a terceira similaridade ocorre às 19:30, onde ambas as entidades estão, ao mesmo tempo, em um mesmo tipo de *POI*, no caso uma sala de aula, embora estejam em atividades diferentes e fisicamente distantes.

Este exemplo demonstra algumas das dificuldades que devem ser tratadas em uma análise de similaridade. Inicialmente, deve ser tratada a questão do alinhamento temporal entre as sequências. Por exemplo, a entidade 'A' poderia ter chegado ao restaurante 10 minutos depois da entidade 'B', mas esta diferença de tempo não influencia na comparação da similaridade neste ponto das duas sequências pois, neste cenário, o fato relevante é que ambas as entidades estiveram próximas, durante um período de tempo e exercendo a mesma atividade.

Ainda considerando-se o tempo, neste exemplo utilizou-se o horário para identificar o momento em que cada contexto foi registrado na sequência. Porém nem sempre este será o valor armazenado nesta dimensão pois, como já explicado na seção 2.2, o conceito de tempo pode ser composto por datas e horas que indiquem um momento específico ou um período de tempo,

Figura 33: Exemplo de comparação de duas sequências de contextos



Fonte: Elaborado pelo Autor

mas também pode ser apenas a ordem relativa em que os eventos ocorreram (DEY; ABOWD; SALBER, 2001). Como o objetivo do *framework* é comparar sequências de contextos, o valor da dimensão tempo possui a função de ser um índice de ordenação das sequências. Isto significa que os contextos das sequências serão comparados na ordem em que ocorreram, com a opção de alinhar as sequências em função do tempo, ou desconsiderar o momento em que os contextos ocorreram e realizar a comparação baseando-se somente na ordem de visitação.

A Tabela 6 demonstra a análise de similaridade entre as sequências ilustradas na Figura 33 aplicando-se o alinhamento temporal, e a Tabela 7 demonstra esta análise considerando-se somente a ordem de visitação de cada contexto. A primeira coluna exibe o valor considerado para ordenação dos contextos. A segunda e a terceira colunas exibem o local e a atividade registrada em cada contexto das sequências A e B respectivamente. A quarta coluna exibe o resultado da análise de similaridade. Uma similaridade parcial pode ocorrer nas seguintes situações:

1. os usuários estão em locais diferentes, mas exercendo a mesma atividade;
2. os usuários estão no mesmo local, mas exercendo atividades diferentes;
3. os usuários estão exercendo a mesma atividade, mas em locais diferentes.

Outra questão apontada por este exemplo é a identificação semântica das posições físicas de cada entidade. Os pontos iniciais e finais de ambas as sequências estão fisicamente distantes

Tabela 6: Análise de similaridade com aplicação de alinhamento temporal

Hora	Entidade A	Entidade B	Similar?
13:00	$[gap]$	Estacionamento D → Estacionando	Não
13:05	$[gap]$	Prédio 6b → Trabalhando	Não
14:55	Estacionamento → Estacionando	Prédio 6b → Trabalhando	Não
15:00	Biblioteca → Estudando	Prédio 6b → Trabalhando	Não
18:20	Atendimento → Em atendimento	Prédio 6b → Trabalhando	Não
19:00	Restaurante → Jantando	Restaurante → Jantando	Sim
19:30	Sala de Aula 1 → Estudando	Sala de Aula 2 → Lecionando	Parcial

Fonte: Elaborado pelo Autor

Tabela 7: Análise de similaridade considerando-se a ordem de visitação

Seq.	Entidade A	Entidade B	Similar?
1	Estacionamento A → Estacionando	Estacionamento D → Estacionando	Parcial
2	Biblioteca → Estudando	Prédio 6b → Trabalhando	Não
3	Atendimento → Em Atendimento	Restaurante → Jantando	Não
4	Restaurante → Jantando	Sala de Aula 2 → Lecionando	Não
5	Sala de Aula 1 → Estudando	$[gap]$	Não

Fonte: Elaborado pelo Autor

e também a identificação de cada localização é diferente. No exemplo, a localização inicial da entidade 'A' é o 'Estacionamento A', enquanto que a localização inicial da entidade 'B' é o 'Estacionamento D'. A utilização de uma ontologia permitiria identificar que ambas as localizações são instâncias de um mesmo conceito, no caso Estacionamentos, o que significaria que neste ponto há um grau de similaridade entre as duas sequências. Porém deve-se considerar que o grau de similaridade neste caso será menor do que seria se ambas as entidades estivessem na mesma localização. O mesmo princípio é valido para as descrições das atividades, ou de qualquer outra informação referente à situação das entidades.

Também deve-se tomar cuidado com a quantidade de variáveis a serem consideradas em uma análise de similaridade. No exemplo acima, além da atividade de cada entidade poderiam ser avaliadas também outras informações, como condições climáticas, sinais vitais ou estado emocional. Porém, acrescentar estes dados à comparação aumentaria a complexidade computacional e não seriam relevantes para os objetivos da análise de similaridade ilustrada neste exemplo. Esta dificuldade pode ser tratada aplicando-se uma etapa de pré-processamento aos dados das sequências analisadas, onde variáveis desnecessárias seriam descartadas.

Por fim, cabe ressaltar que este é somente um dos possíveis exemplos de sequências de contextos que o *framework* pretende tratar. Lembrando a definição de DEY; ABOWD; SALBER (2001), de que contexto é qualquer informação que possa ser usada para caracterizar a situação de entidades, uma pessoa é apenas um dos tipos de entidades cujas sequências podem ser analisadas. Neste mesmo cenário descrito no exemplo é possível identificar outras entidades potencialmente relevantes para alguma aplicação. Por exemplo, o restaurante em que ambos

os usuários jantaram é, por si só, uma entidade. Por se tratar de uma entidade imóvel sua sequência de contextos não apresentaria modificações quanto a Localização, porém registraria alterações na Situação do restaurante ao longo do tempo. Esta informação seria coletada através de sensores que poderiam trazer dados como a temperatura ambiente, condições climáticas ou o nível de ruído, ou ainda extraídas a partir de bases de dados, como a quantidade de clientes em determinado horário, os produtos vendidos e o nível de satisfação dos clientes. Uma análise de similaridade neste tipo de sequência poderia ser utilizada para encontrar semelhanças com outros restaurantes, identificar padrões de comportamento de clientes, entre outras informações.

Poderão ainda haver casos onde as variáveis relevantes serão compostas somente por números. Nestes casos, conforme descrito na seção 2.6, é possível tratar as sequências de contextos como um tipo de série temporal. Portanto o *SIMCOP* deve oferecer suporte às técnicas de analise de similaridade entre séries temporais e também oferecer funcionalidades para cálculos estatísticos sobre as sequências analisadas.

4.2 Requisitos

4.2.1 Requisitos Funcionais (RF)

Para que o *Framework* atendesse aos objetivos expostos, são especificadas as seguintes funcionalidades:

1. Análise de Similaridade
   (a) Calcular a similaridade entre dois contextos;
   (b) Calcular a similaridade global de duas sequências de contextos;
   (c) Permitir a utilização combinada de diferentes métricas de similaridade, conforme os tipos de informações disponíveis nos contextos analisados.
2. Configurabilidade e Extensibilidade
   (a) Permitir a configuração dos parâmetros operacionais da análise de similaridade;
   (b) Facilitar a implementação de novas métricas de similaridade, através da extensão de classes existentes ou implementação de novas classes a partir das interfaces oferecidas;
   (c) Permitir configurar em quais situações cada ontologia deve ser utilizada;
3. Filtros e Transformações
   (a) Permitir a aplicação de filtros nos dados de entrada e de saída para aplicação de critérios de seleção ou realização de procedimentos adicionais, como por exemplo, registro de *logs*;

(b) Permitir a aplicação de transformações nos dados de entrada e de saída, para realização de procedimentos de pré e pós processamento, como normalização dos dados, redução de dimensionalidade e análise de atributos globais das sequências de contextos;

(c) Permitir definir a ordem em que os filtros e transformações devem ser aplicados.

4. Acesso a dados

   (a) Permitir o armazenamento e consulta de sequências de contextos em meios externos como bancos de dados ou arquivos;

   (b) Permitir o armazenamento e consulta de especificações de ontologias nestes mesmos meios.

4.2.2 Requisitos Não Funcionais (RNF)

Segundo PRESSMAN (2006), requisitos não funcionais são aqueles que estabelecem a qualidade e as restrições sobre os serviços oferecidos por um *software*. Para os objetivos deste trabalho, são considerados relevantes os seguintes RNF's:

1. **Requisitos de Interoperabilidade**: O conceito de *framework* implica necessariamente em uma especificação que facilite o acoplamento aos mais diversos tipos de aplicações. Isto significa primeiramente que o *framework* deve especificar de forma clara e não ambígua, os procedimentos que a aplicação deve adotar para consumir os serviços oferecidos, o formato esperado dos dados de entrada e a formalização dos resultados gerados. Por outro lado, o *framework* deve oferecer meios de consumo de seus serviços que sejam compatíveis com as tecnologias adotadas pela aplicação;

2. **Requisitos de Padrões**: Para facilitar a manutenção e a extensão do *Framework*, é necessário que tanto a especificação quanto a implementação sigam um padrão coerente de implementação. O desenvolvimento deste trabalho irá utilizar os seguintes padrões:

   (a) Programação Orientada a Objetos.

   (b) Design Pattern: Singleton [1]

   (c) Design Pattern: Factory [2]

3. **Requisitos de Eficiência**: O *Framework* deve otimizar suas operações para atingir, quando possível, uma complexidade de tempo próxima a $O(n)$. Em alguns casos, como por exemplo nos algoritmos de busca em ontologias, poderão ser configurados limitadores e outros controles para garantir que o processo não fique executando indefinidamente.

[1] http://www.oodesign.com/singleton-pattern.html
[2] http://www.oodesign.com/factory-pattern.html

4. **Requisitos de Portabilidade**: O *Framework* deverá oferecer suporte ao desenvolvimento de aplicações em diversas plataformas, como *desktops*, aplicações WEB e dispositivos móveis.

5. **Requisitos de Implementação**: O *Framework* será implementado inicialmente em linguagem JAVA[3].

4.2.3 Premissas

A representação interna dos dados contextuais, no *framework*, é baseada na linguagem SituationML (HECKMANN, 2005). Os dados que a aplicação gerencia devem estar próximos a este formato ou devem poder ser convertidos, utilizando-se as ferramentas oferecidas pelo *framework*.

O *framework* pode fazer uso de bibliotecas de terceiros para implementar funcionalidades específicas, como por exemplo a interação e inferência em ontologias ou determinadas métricas de similaridade. O *framework* oferecerá classes chamadas de Adaptadoras para comunicar-se com estas bibliotecas, permitindo que usuários possam expandir este aspecto do *framework*.

4.3 Especificação

4.3.1 Diagrama de domínio

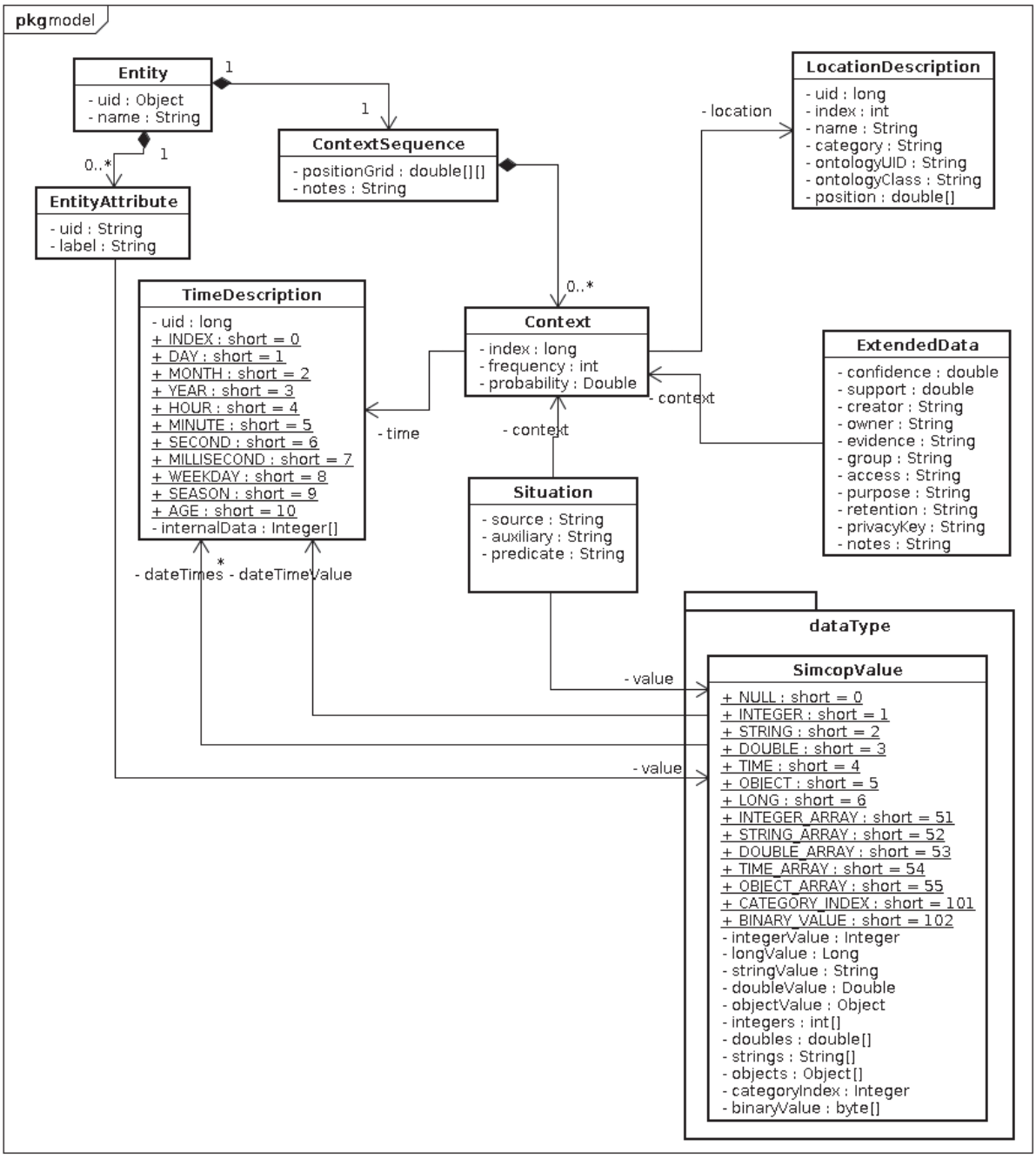
A Figura 34 exibe o diagrama de domínio do *SIMCOP*. Cada sequência de contextos pertence a uma entidade, que deve ter uma identificação única durante o processo de análise de similaridade, e é descrita por um nome e uma lista de atributos. A sequência é composta por registros de contextos, cronologicamente ordenados. Isto não significa que é necessário identificar a data e hora, mas sim que a ordem em que cada contexto ocorreu deve ser preservada. Cada registro pode descrever os dados contextuais em quatro dimensões:

1. ***TimeDescription***: Descreve o tempo informado nos campos *start* e *end* do contexto;
2. ***Location***: Identifica a posição geográfica (latitude, longitude e elevação) e o nome e/ou categoria da localização (Casa, Trabalho, Escola, entre outros);
3. ***Situation***: Composto por uma lista de variáveis que descrevem o estado, situação ou atividade da entidade no momento e no local apontados por este registro;
4. ***ExtendedData***: Contém informações adicionais que podem ser úteis para alguns tipos de análise.

A Tabela 8 descreve os conceitos expostos neste diagrama.

[3]http://www.oracle.com/technetwork/java/javase/index.html

Figura 34: Diagrama de Domínio do *SIMCOP*



Fonte: Elaborado pelo autor

4.3.2 Diagrama de Casos de Uso

A Figura 35 descreve os casos de uso do *framework*. Os desenvolvedores da aplicação cliente devem inicialmente desenvolver uma classe que implemente a interface *ISequenceSource*. Esta classe será responsável por acessar os dados originais da aplicação, independente do local onde estes dados estiverem armazenados, e então representá-los através de instâncias das classes que compõem a representação interna dos contextos, demonstradas na Figura 34.

Tabela 8: Descrição de conceitos de domínio do *framework*

Conceito	SituationML	Descrição
Entity	*Subject*	Entidade proprietária da sequência de contextos
EntityAttribute	—	Lista de atributos adicionais da entidade
ContextSequence	—	Contém a lista de registros contextuais da entidade, cronologicamente ordenados
Context	*SituationReport*	Contém a descrição da situação da entidade, em um momento e local específico
TimeDescription	*Start, End*	Armazena informações adicionais em relação ao tempo apontado nos campos *start* e *end* do registro de contexto
Location	*Location, Position*	Identifica a posição geográfica da entidade no momento deste registro e descreve a localização que esta posição representa
Situation	*Source, MainPart*	Descreve a situação de um contexto em uma lista de afirmações do tipo **predicado=objeto**. Um predicado pode ser complementado por um atributo *auxiliary*, que pode ser usado para resolver ambiguidades. Também é possível identificar a origem (*source*) de cada predicado, por exemplo a identificação do sensor que fez a leitura de determinado valor (*object*).
ExtendedData	*Administration, Privacy*	Informações adicionais sobre o contexto
SIMCOPValue	—	Esta classe tem como função encapsular os valores dos predicados (*object*), preservando o tipo de dado e evitando, desta forma, a necessidade de conversões de tipos durante a análise de similaridade.

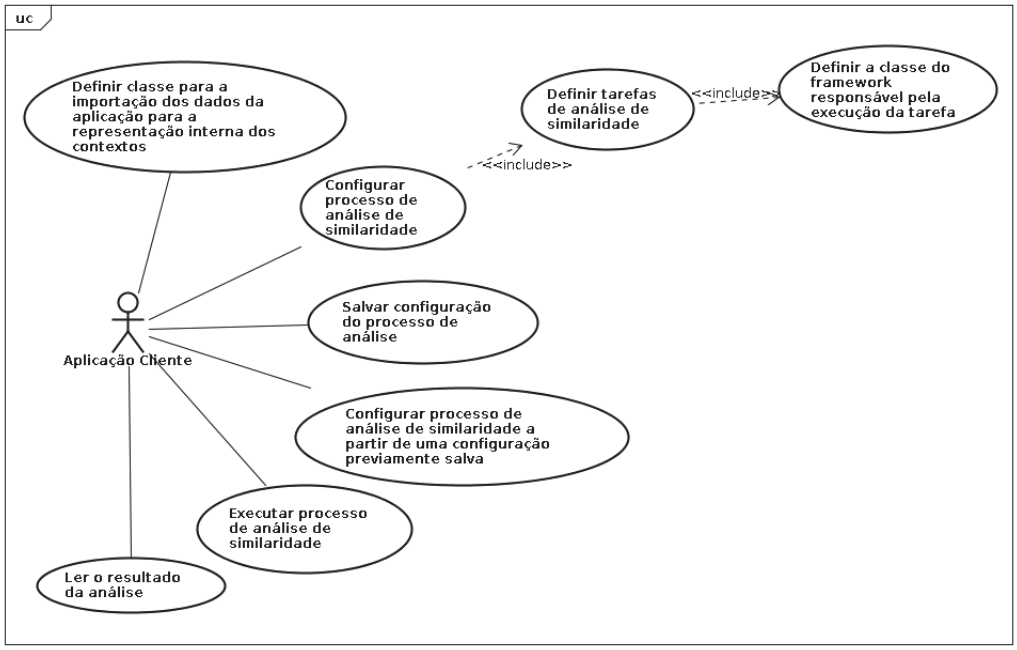
Fonte: Elaborado pelo Autor

O caso de uso seguinte consiste em configurar o processo de análise de similaridade. O *framework* considera um processo de análise como sendo uma sequência de tarefas (*tasks*) que constituem um processo de mineração de dados composto pelas etapas de pré-processamento, mineração e pós-processamento conforme HAN; KAMBER (2006). A função de cada *task* é implementada por uma classe do *framework*, ou pode ser implementada pela aplicação, estendendo uma das classes do *framework* se o desenvolvedor da aplicação assim desejar.

O *framework* prevê ainda que um processo de análise possa ser salvo em algum meio de armazenamento externo, de modo que o processo ser reutilizado em diversas aplicações. São disponibilizadas classes para salvar os processos em arquivos XML, bem como classes abstratas que permitirão o armazenamento destes em outros meios, como por exemplo, em um banco de dados.

Uma vez definido o processo de análise, o mesmo pode ser executado invocando-se o *kernel* do *framework*, implementado na classe *SIMCOP*. O *kernel* recebe como parâmetro uma ins-

Figura 35: Diagrama de Casos de Uso do *Framework*



Fonte: Elaborado pelo autor

tância do processo a ser executado e duas sequências de contextos, sobre as quais o processo é executado.

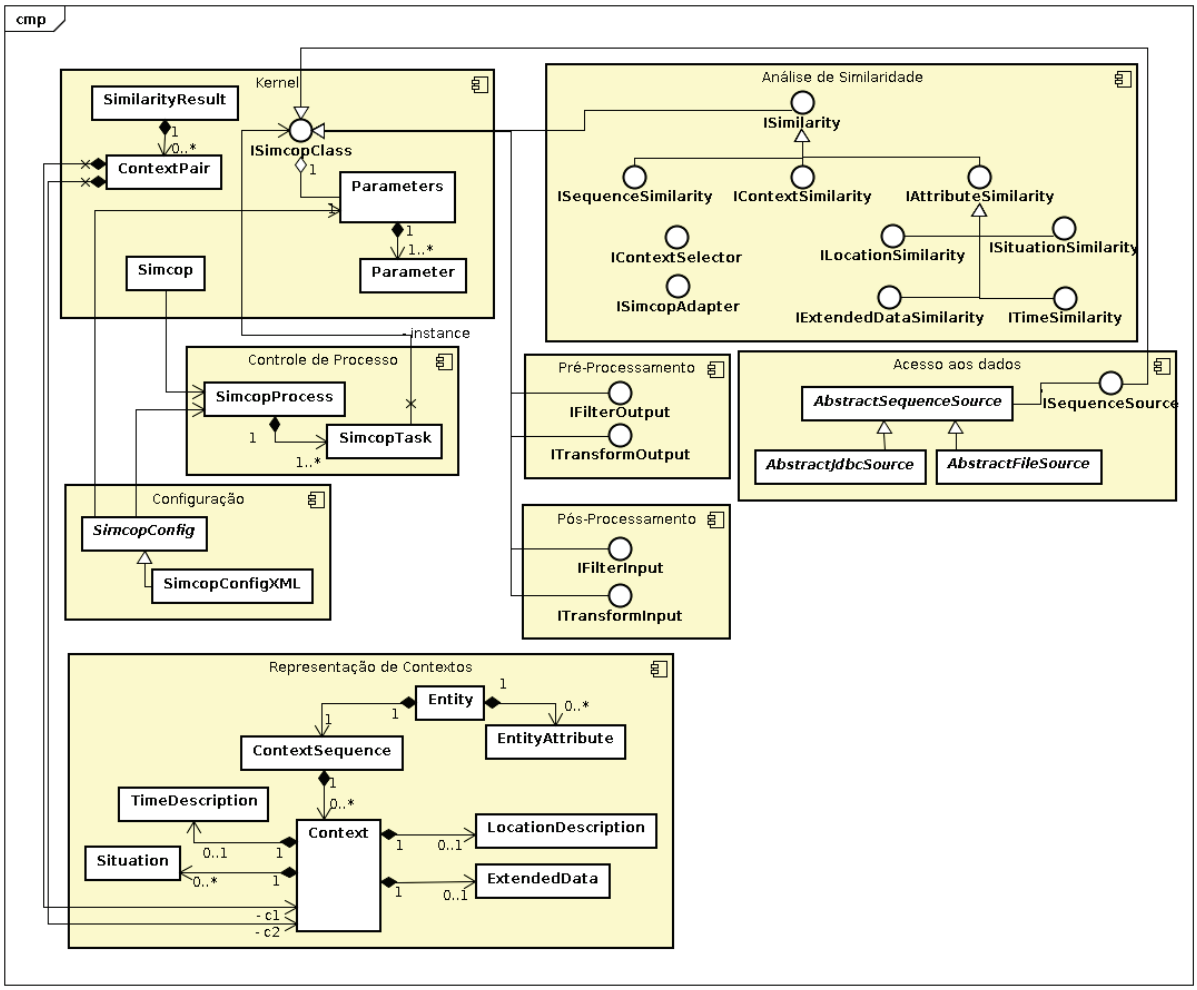
Por fim, a aplicação recebe do *kernel* uma instância da classe *SimilarityResult* que contém um valor numérico correspondente ao valor da similaridade ou distância calculada pelo processo e uma sequência de pares de contexto, onde cada par é acompanhado do valor da similaridade ou distância calculado entre os contextos do par.

4.3.3 Componentes

A Figura 36 ilustra os componentes do *framework*, responsáveis por fornecer a interface para as aplicações clientes, gerenciar a configuração de parâmetros operacionais, carregar dados das sequências de contextos, executar tarefas de pré e pós processamento e executar a análise da similaridade.

No componente ***KERNEL*** está localizada a classe *SIMCOP* que deverá ser chamada pela aplicação cliente, para executar a análise de similaridade sobre dados carregados através de um *Sequence Source* e representados como sequências de contextos. O processo de análise é executado comparando-se duas sequências, e o resultado será uma instância da classe *SimilarityResult*, composta por um conjunto de pares de contextos (classe *ContextPair*) considerados similares acompanhado de um valor numérico, que representa o valor da similaridade ou distância entre os contextos, calculado pela função de análise de similaridade de contextos (*IContextSimilarity*) escolhida. A classe *SimilarityResult* também armazena um valor numérico que representa o valor global da similaridade ou distância entre duas sequências, calculado

Figura 36: Componentes do *Framework*



Fonte: Elaborado pelo autor

pela função de similaridade de sequências (*ISequenceSimilarity*) escolhida. Por fim a interface *ISIMCOPClass* determina o comportamento de todas as classes configuráveis do sistema. As configurações necessárias a cada classe são armazenadas em um conjunto de parâmetros, sendo que cada parâmetro é composto por uma chave *"key"="value"*, onde *"value"* e *"key"* são *Strings* possíveis de serem armazenadas em algum meio físico, como por exemplo, um arquivo de configuração.

A classe *SIMCOP* é responsável por executar um processo de análise de similaridade, que deverá ser especificado usando-se as classes do componente **Controle de Processo**. Este é constituido pela classe *SIMCOPProcess*, que por sua vez é composta de uma sequência ordenada de tarefas (*SIMCOPTask*). Cada tarefa especifica o nome da classe de uma das interfaces do *framework*, e o conjunto de parâmetros que deverá ser carregado na instância da classe especificada ao executar a tarefa.

Conforme descrito na seção 4.3.2, o *SIMCOP* trata o processo de análise de similaridade como um processo de mineração de dados (HAN; KAMBER, 2006). Os componentes descritos a seguir são responsáveis por implementar cada etapa do processo de mineração:

1. **Seleção**, implementada pelo componente de **Acesso aos Dados**, é responsável por acessar os dados originais da aplicação e carregá-los em instâncias das classes que constituem o componente **Representação de Contextos**. Este componente é o principal *hot spot* (PREE, 1994) do *framework*, uma vez que somente a aplicação cliente possuirá o conhecimento para acessar seus dados e convertê-los em sequências de contextos;

2. **Pré-Processamento** e **Transformação**, implementadas pelo componente de **Pré- Processamento**, que é composto pelas interfaces *IInputFilter* e *IInputTransformation*. As classes que implementam estas interfaces serão as primeiras da sequência de tarefas que irão compor o processo de análise de similaridade, e serão portanto, executadas antes da análise de similaridade, visando a preparação inicial dos dados. O *framework* já implementa algumas classes capazes de realizar filtros básicos, baseados no conteúdo dos contextos, e também transformações como o alinhamento dos contextos de duas sequências em função do tempo. Entretanto classes mais sofisticadas poderão ser implementadas usando-se as interfaces deste componente, tornando-o também um *hot spot*;

3. **Mineração de dados**, implementada pelo componente de **Análise de Similaridade**, onde a classe selecionada para esta tarefa deve implementar a interface *ISequenceSimilarity*. Esta classe pode implementar internamente seu próprio método de comparação de contextos ou pode exigir que seja especificada uma classe que implemente a interface *IContextSimilarity* para realizar calcular a similaridade entre contextos. Da mesma forma, a comparação entre os atributos dos contextos, como situação, localização e tempo, pode ser implementada diretamente pela classe de similaridade de contextos, ou pode exigir que seja especificada uma classe para comparação de cada tipo de atributo presente. Uma classe que implemente a interface *IContextSelector* é então utilizada após a comparação, para avaliar se dois contextos avaliados atenderam aos critérios especificados para serem considerados similares. Por fim, a interface *ISIMCOPAdapter* fornece o mecanismo através do qual será possível acoplar bibliotecas de terceiros às classes de análise de similaridade, pela implementação de um adaptador entre as interfaces do *framework* e os métodos da biblioteca externa;

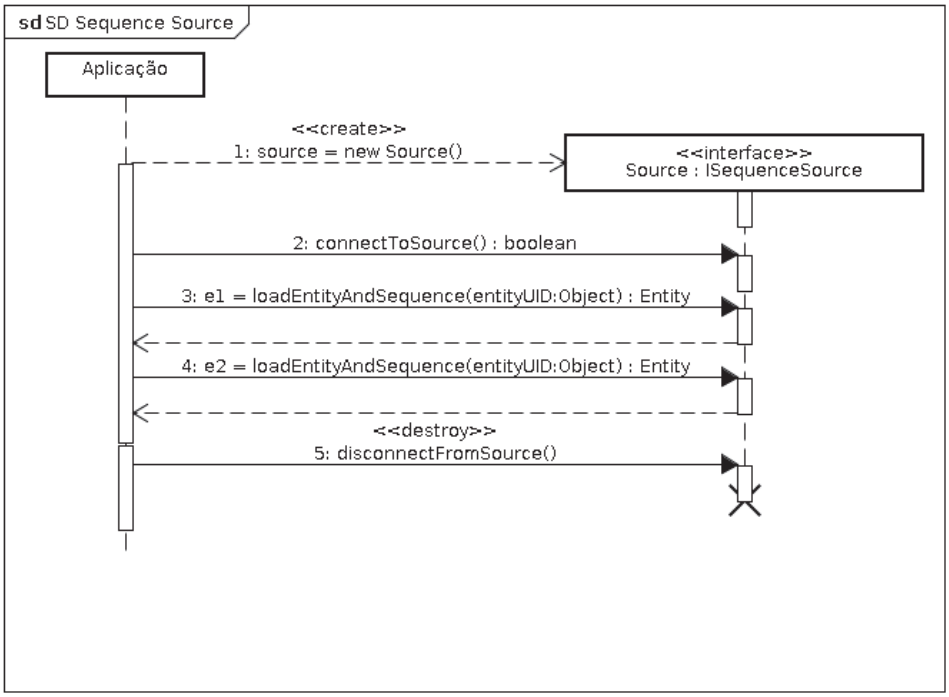
4. **Pós-Processamento** Implementado pelo componente homônimo, possui as mesmas funcionalidades do componente de pré-processamento com a diferença de que as interfaces deste componente atuam sobre o resultado da análise, podendo-se aplicar filtros ou transformações sobre os pares de contextos que foram considerados similares;

5. **Interpretação e Avaliação** Por fim, cabe a aplicação cliente ler e interpretar os dados retornados pelo *framework* através de uma instância da classe *SIMCOPResult*.

4.3.4 Diagramas de Sequência

A Figura 37 ilustra o funcionamento das classes responsáveis por executar a seleção e carga das sequências de contextos. A aplicação deve implementar uma classe que implemente a interface *ISequenceSource*, podendo opcionalmente extender uma das classes abstratas disponíveis no componente, com o objetivo de aproveitar o código já existente. Esta interface especifica cinco métodos que devem ser implementados: *connectToSource()*, responsável por acessar o meio físico onde os dados da aplicação estão armazenados, como por exemplo em um banco de dados; *getListEntities(Object... queryParameters)*, responsável por carregar a lista de entidades disponíveis no meio físico, considerando os critérios de pesquisa recebidos por parâmetro; *loadEntity(Object uid)*, responsável por carregar os dados o os atributos de uma entidade específica; *loadIntoSequence(ContextSequence sequence)*, responsável por carregar a sequência de contextos de uma entidade; e *disconnectFromSource()*, responsável por desfazer a conexão com o meio físico de armazenamento dos dados. A aplicação executa então a carga do contextos, abrindo uma conexão ao meio físico e invocando o método *loadEntityAndSequence(Object entityUID)*, que retornará os contextos de uma entidade conforme a implementação dos métodos *loadEntity* e *loadIntoSequence*.

Figura 37: Diagrama de Sequência do *Sequence Source*

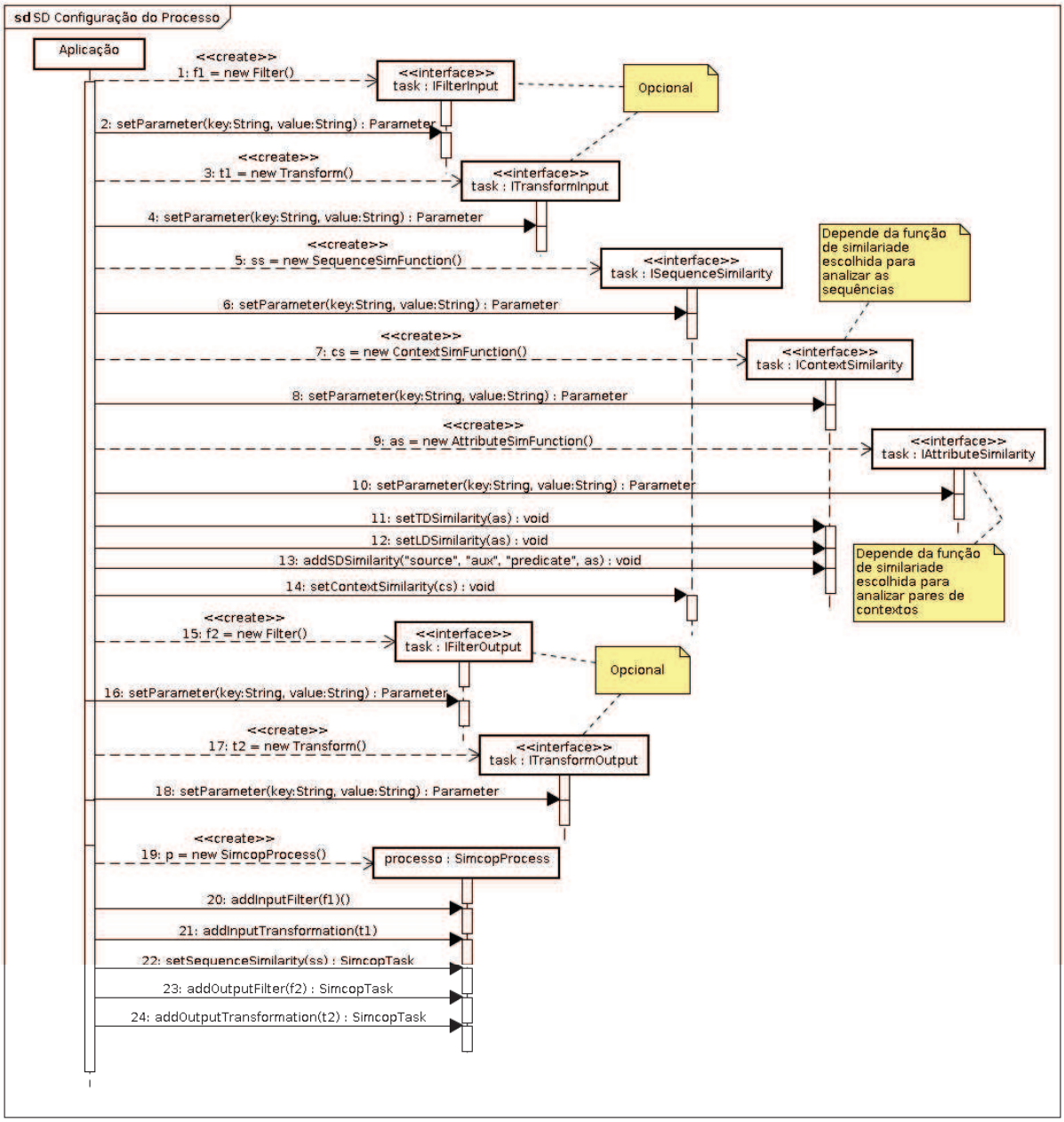


Fonte: Elaborado pelo Autor

A Figura 38 especifica como um processo de análise de similaridade deve ser configurado via código fonte. Cada tarefa recebe uma instância de uma classe que implemente uma das interfaces do componente de análise de similaridade, já devidamente configurada através de

seus parâmetros. Em seguida, cada tarefa é adicionada ao processo, invocando-se o método relativo ao tipo de tarefa criada.

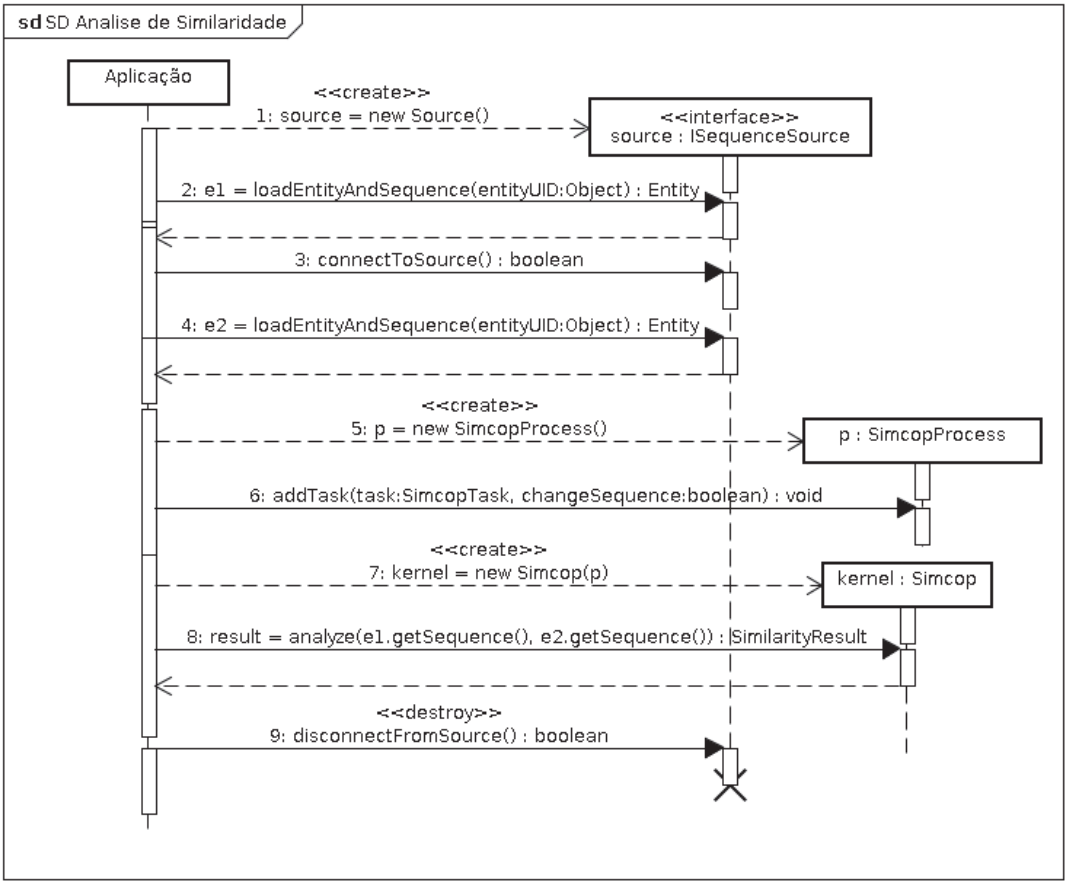
Figura 38: Diagrama de Sequência da configuração do processo



Fonte: Elaborado pelo Autor

A Figura 39 ilustra a utilização do *SIMCOP* para análise de similaridade entre duas sequências de contextos. Inicialmente, carrega-se os dados dos contextos através de um ou dois *Sequence Source*s, configura-se o processo de análise e executa-se a análise usando a classe *SIMCOP* do componente *kernel*. O resultado é recebido na forma de uma instância da classe *SimilarityResult*, que contém o valor da similaridade ou distância entre as sequências calculado, os pares de contextos similares e o valor de similaridade ou distância de cada par.

Figura 39: Diagrama de Sequência da análise de similaridade



Fonte: Elaborado pelo Autor

4.4 Considerações sobre o capítulo

Neste capítulo foi apresentado a especificação do *SIMCOP*. Inicialmente foram apresentados os conceitos fundamentais adotados no *framework* e um exemplo de utilização destes conceitos. Em seguida foi realizado um levantamento de requisitos, procurando apontar as funcionalidades que deveriam ser atendidas pela especficiação. A partir deste levantamento foi criada a especificação do *framework*, abordando o modelo de dados utilizado internamente baseado nas definições de DEY; ABOWD; SALBER (2001) e de HECKMANN (2005) para formalização de contextos, os componentes e *interfaces* do qual o *framework* dispõe, e a ilustração do processo de análise de similaridade através dos diagramas de sequência.

5 ASPECTOS DE IMPLEMENTAÇÃO E AVALIAÇÃO

Para avaliação do *framework* proposto na especificação do *SIMCOP* foi implementado um protótipo, em linguagem JAVA, com o qual foram desenvolvidas duas aplicações voltadas para recomendação de itens a partir da similaridade entre históricos de contextos. O objetivo do desenvolvimento destas aplicações foi avaliar a eficácia da análise de similaridade, bem como a utilidade do *framework*, em situações reais usando como critério a relevância das recomendações, avaliadas de forma subjetiva pelos usuários e objetiva através da verificação de semelhanças entre os objetos recomendados.

Este capítulo está dividido em quatro seções. A primeira seção detalha a implementação do protótipo. A segunda seção descreve o componente de configuração, que permite definir processos de análise de similaridade independentes da aplicação. A terceira seção descreve as aplicações que foram desenvolvidas para avaliação do *framework*: o ReBaSS, um sistema de recomendação de objetos de aprendizagem, e o uso do SIMCOP em um sistema de experimental de filtragem colaborativa para recomendação de livros em uma biblioteca, denominado *U-Library*. Por fim, na quarta seção são apresentadas as considerações finais deste capítulo.

5.1 Implementação do protótipo

Inicialmente as *interfaces* especificadas pelo *framework* foram escritas em linguagem JAVA. Algumas funções básicas, de uso geral, foram implementadas em classes abstratas que implementam as respectivas *interfaces*. Diferentes funções de análise de similaridade foram implentadas na forma de classes concretas, que por sua vez podem ser utilizadas em processo de análise de similaridade, representado pela classe *SIMCOPProcess*.

A *interface ISimilarity* possui três *interfaces* herdeiras, cada uma responsável por definir os métodos das classes que implementam as funções de análise de similaridade para cada nível de análise: similaridade entre sequências de contextos, similaridade entre contextos e similaridade entre atributos de contextos. Para a análise de similaridade entre sequências de contextos o *Framework* prevê dois tipos básicos de análise, definidos pelas classes abstratas *EachContext* e *WholeSequences*. A classe *EachContext*, descrita na Listagem 1, percorre cada contexto da sequência *A* (*ctxA*) e compara-o com o contexto na sequência *B* (*ctxB*) que esteja na mesma posição do que o contexto *ctxA*, aplicando a função de análise de similaridade entre contextos especificada pelo processo de análise. A função de similaridade entre contextos retorna então um valor que é passado a uma instância da interface *IContextSelector*, responsável por decidir se o par de contextos deverá fazer parte do resultado final. Caso a firmativo, esta comparação gera um objeto *ContextPair* composto pelos dois contextos avaliados, acompanhados de um valor numérico que indica a similaridade ou distância entre os pares, conforme a função de análise de similaridade entre contextos definida.

Listagem 1: Função de Análise de Similaridade *EachContext*

```
public abstract class EachContext extends SequenceSimilarity {
    public boolean isContextSimilarityNeeded() { return true; }
    protected SimilarityResult internalGetSimilarity(ContextSequence s1, ContextSequence s2) throws Exception {
        SimilarityResult result = new SimilarityResult(s1,s2);
        if (s1 != null && s2 != null) {
            IContextSelector ctxSelector = createContextSelector();
            if (contextSimilarity != null) {
               contextSimilarity.validateParameters();
            }
            for (int index = 0; index < Math.max(s1.size(), s2.size()); index++) {
                Context ctxA = index < s1.size() ? s1.get(index) : null;
                Context ctxB = index < s2.size() ? s2.get(index) : null;
                evaluate(ctxA, ctxB, ctxSelector, result);
            }
            setSequenceSimilarityValue(s1, s2, result);
        }
        return result;
    }
    protected void evaluate(Context ctxA, Context ctxB, IContextSelector contextSelector, SimilarityResult result)
                    throws Exception {
        double similarityOrDistance = contextSimilarity.getSimilarity(ctxA, ctxB);
        boolean accepted;
        if (contextSelector.isSelectContext(similarityOrDistance, contextSimilarity, ctxA, ctxB)) {
            result.add(ctxA, ctxB, similarityOrDistance);
            accepted = true;
        } else {
            accepted = false;
        }
        afterEvaluate(ctxA, ctxB, contextSelector, result, similarityOrDistance, accepted);
    }
    protected void afterEvaluate(Context ctxA, Context ctxB,
                                 IContextSelector contextSelector, SimilarityResult result,
                                 double similarityOrDistance, boolean contextAccepted)
    {    }
    protected abstract void setSequenceSimilarityValue(ContextSequence s1, ContextSequence s2,
                                                       SimilarityResult result);
    protected abstract IContextSelector createContextSelector();
}
```

Já a classe *WholeSequences* descrita na Listagem 2, engloba funções de análise de similaridade que analisam ambas as sequências como dois conjuntos, não necessariamente realizando uma análise iterativa, como no caso da *EachContext*. São exemplos de funções deste tipo as técnicas de análise de similaridade que comparam trajetórias de entidades, ou que tratem as sequências de contextos como séries temporais.

Listagem 2: Função de Análise de Similaridade *WholeSequences*

```
public abstract class WholeSequences extends SequenceSimilarity {
    protected Double[][] buildMatrix(ContextSequence s1, ContextSequence s2, ContextSimilarity f)
                        throws Exception {
        Double[][] result = new Double[s1.size()][s2.size()];
        f.validateParameters();
        for (int i = 0; i < s1.size(); i++) {
            for (int j = 0; j < s2.size(); j++) {
                Context c1 = s1.get(i);
                Context c2 = s2.get(j);
                result[i][j] = f.getSimilarity(c1, c2);
            }
        }
        return result;
    }
    protected Boolean[][] buildEqualsMatrix(ContextSequence s1, ContextSequence s2) {
        Boolean[][] result = new Boolean[s1.size()][s2.size()];
        for (int i = 0; i < s1.size(); i++) {
            for (int j = 0; j < s2.size(); j++) {
                Context c1 = s1.get(i);
                Context c2 = s2.get(j);
                result[i][j] = c1 == null ? c2 == null : c1.equals(c2);
            }
        }
        return result;
    }
}
```

Um exemplo de implementação de uma função de análise de similaridade para sequências de contextos do tipo *EachContext* pode ser verificado na Listagem 3. Nesta Listagem está a implementação da função de similaridade *ECDefault*, que compara cada contexto da sequência *A* com um contexto da sequência *B* que esteja na mesma posição, e retorna uma lista de pares de contextos (*ContextPair*) que atingiram o valor de similaridade minimo ou de distância máxima recebido através dos parâmetros *MAX_DISTANCE* e *MIN_SIMILARITY*, ou retornando todos os pares caso o parâmetro *ACCEPT_ALL* seja igual a "*true*". Por fim, os parâmetros *ACCEPT_GAPS_SEQUENCE_** indicam o tratamento que deve ser adotado quando um contexto em uma das sequências em análise não possuir correspondente, na mesma posição, na outra sequência.

Listagem 3: Exemplo de uma função de similaridade entre sequências

```
/**
 * Compares contexts in sequences A and B, following respectives indexes and returning pairs {Ca, Cb} of
 * contexts that have reached the minimum similarity value, calculated by the object {@link #contextSimilarity}.
 */
public class ECDefault extends EachContext {
    public static final String ACCEPT_ALL = "acceptAll";
    public static final String MAX_DISTANCE = "maxDistance";
    public static final String MIN_SIMILARITY = "minSimilarity";
    public static final String ACCEPT_GAPS_SEQUENCE_A = "acceptGapsSequenceA";
    public static final String ACCEPT_GAPS_SEQUENCE_B = "acceptGapsSequenceB";
    private boolean acceptAll, acceptGapsA, acceptGapsB, maxDistance, minSimilarity, similaritySum;
    protected SimilarityResult internalGetSimilarity(ContextSequence s1, ContextSequence s2) throws Exception {
        this.similaritySum = 0;
        return super.internalGetSimilarity(s1, s2);
    }
    protected void afterEvaluate(Context ctxA, Context ctxB, IContextSelector ctxSelector,
                                 SimilarityResult result, double simOrDistance, boolean ctxAccepted) {
        this.similaritySum += simOrDistance;
    }
    public Parameters getDefaultParameters() {
        Parameters pars = new Parameters();
        pars.addParameter(ACCEPT_ALL, "false");
        pars.addParameter(MAX_DISTANCE, "1");
        pars.addParameter(MIN_SIMILARITY, "0.01");
        pars.addParameter(ACCEPT_GAPS_SEQUENCE_A, "false");
        pars.addParameter(ACCEPT_GAPS_SEQUENCE_B, "false");
        return pars;
    }
    protected void internalValidateParameters() throws Exception {
        acceptAll = "true".equalsIgnoreCase(getSimpleParameter(ACCEPT_ALL));
        acceptGapsA = "true".equalsIgnoreCase(getSimpleParameter(ACCEPT_GAPS_SEQUENCE_A));
        acceptGapsB = "true".equalsIgnoreCase(getSimpleParameter(ACCEPT_GAPS_SEQUENCE_B));
        maxDistance = Double.parseDouble(getSimpleParameter(MAX_DISTANCE).trim());
        minSimilarity = Double.parseDouble(getSimpleParameter(MIN_SIMILARITY).trim());
    }
    protected IContextSelector createContextSelector() {
        if (contextSelector == null) {
            DefaultContextSelector result = new DefaultContextSelector();
            result.setAcceptAll(acceptAll);
            result.setAcceptGapsA(acceptGapsA);
            result.setAcceptGapsB(acceptGapsB);
            result.setMaxDistance(maxDistance);
            result.setMinSimilarity(minSimilarity);
            return result;
        } else {
            return contextSelector;
        }
    }
    protected void setSequenceSimilarityValue(ContextSequence s1, ContextSequence s2, SimilarityResult result) {
        double maxCtx = getContextSimilarity().getLargestPossibleValue();
        double maxSimilarity = Math.max(s1.size() * maxCtx, s2.size() * maxCtx);
        result.setCalculatedValue( maxSimilarity != 0 ? similaritySum / maxSimilarity : similaritySum );
    }
    public boolean isDistanceFunction() {
        return contextSimilarity == null ? false : contextSimilarity.isDistanceFunction();
    }
}
```

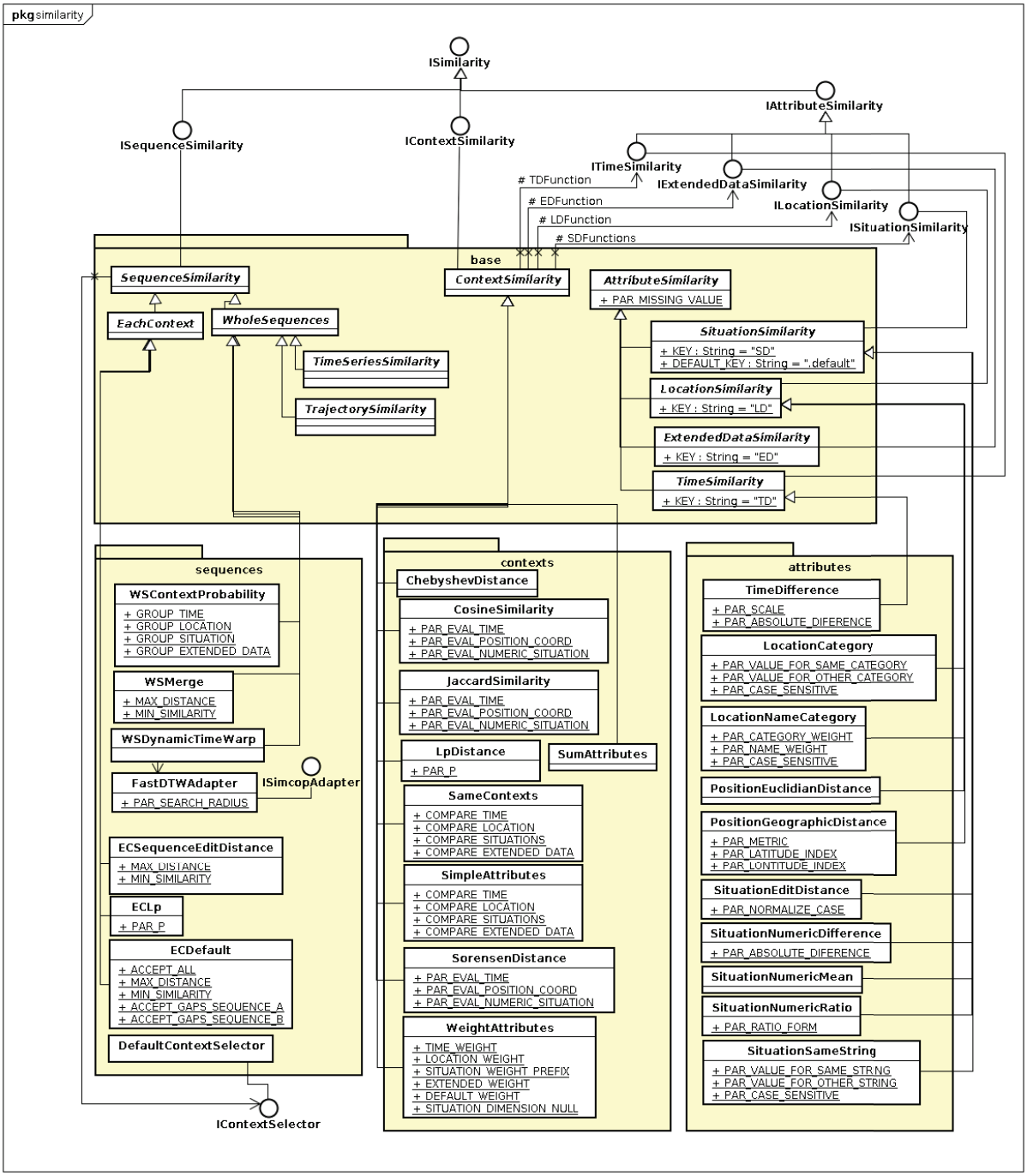
Ainda na função *ECDefault*, a comparação entre dois contextos em um par é realizada utilizando-se uma função de análise de similaridade entre contextos, que por sua vez pode ser qualquer uma das funções de análise disponíveis no *framework*, como por exemplo, a Distância do Cosseno cuja implementação pode ser vista na Listagem 4, e que considera todos os atributos numéricos de um contexto como coordenadas de um vetor em um espaço N-dimensional, sendo que a distância entre dois contextos é dada pelo cosseno do ângulo entre estes dois vetores.

Listagem 4: Exemplo de uma função de similaridade entre contextos

```
/** Implements the Cosine Similarity function. */
public class CosineSimilarity extends ContextSimilarity {
    public static final String PAR_EVAL_TIME = "evaluateTime";
    public static final String PAR_EVAL_POSITION_COORD = "evaluatePositionCoordinates";
    public static final String PAR_EVAL_NUMERIC_SITUATION = "evaluateNumericValueSituations";
    private boolean evalTime = false;
    private boolean evalPositionCoord = false;
    private boolean evalNumericSituations = true;
    public boolean isDistanceFunction() { return false; //is a Smilarity function }
    public boolean isAttributeSimilarityNeeded() { return false; }
    public double getLargestPossibleValue() { return 1; }
    public Parameters getDefaultParameters() {
        Parameters pars = new Parameters();
        pars.addParameter(PAR_EVAL_TIME, "false");
        pars.addParameter(PAR_EVAL_POSITION_COORD, "false");
        pars.addParameter(PAR_EVAL_NUMERIC_SITUATION, "true");
        return pars;
    }
    protected void internalValidateParameters() throws Exception {
        evalTime = "true".equalsIgnoreCase( getSimpleParameter(PAR_EVAL_TIME) );
        evalPositionCoord = "true".equalsIgnoreCase( getSimpleParameter(PAR_EVAL_POSITION_COORD) );
        evalNumericSituations = "true".equalsIgnoreCase( getSimpleParameter(PAR_EVAL_NUMERIC_SITUATION) );
    }
    protected double internalGetSimilarity(Context c1, Context c2) throws Exception {
        List<double[]> map = getAttributesAsCoord(c1, c2, evalTime, evalPositionCoord, evalNumericSituations);
        double[] P = new double[map.size()];
        double[] Q = new double[map.size()];
        for (int i = 0; i < map.size(); i++) {
            double[] PQ = map.get(i);
            P[i] = PQ[0];
            Q[i] = PQ[1];
        }
        return CommonFunctions.cosineSimilarity(P, Q);
    }
    protected List<double[]> getAttributesAsCoord(Context c1, Context c2, boolean time, boolean position, boolean situations)
                            throws Exception {
        if (c1 == null || c2 == null) { return null; }
        List<double[]> result = new ArrayList<double[]>();
        if (time) {
            Long v1 = c1.getTime() != null ? c1.getTime().asLong() : null;
            Long v2 = c2.getTime() != null ? c2.getTime().asLong() : null;
            result.add( new double[] { v1 != null ? v1 : 0,  v2 != null ? v2 : 0} );
        }
        if (position) {
            double[] coordA = c1.getLocation() != null ? c1.getLocation().getPosition() : null;
            double[] coordB = c2.getLocation() != null ? c2.getLocation().getPosition() : null;
            for (int i = 0; i < coordA.length; i++) {
                result.add( new double[] { coordA[i], coordB[i]} );
            }
        }
        if (situations) {
            Map<String, Situation[]> allSituations = unionSituations(c1, c2);
            for (String sitKey : allSituations.keySet()) {
                Situation[] pair = allSituations.get(sitKey);
                Situation sitA = pair[0];
                Situation sitB = pair[1];
                Number valueA = Utils.getAsNumber(sitA.getValue());
                Number valueB = Utils.getAsNumber(sitB.getValue());
                result.add( new double[] { valueA.doubleValue(), valueB.doubleValue()} );
            }
        }
        return result;
    }
}
```

A Figura 40 mostra a estrutura completa de classes para análise de similaridade implementadas no protótipo, bem como as constantes que representam os nomes dos parâmetros operacionais de cada classe. O pacote *base* contém as classes abstratas, enquanto que os pacotes *sequences*, *contexts* e *attributes* contém as implementações para as funções análise de similaridade entre sequências de contextos, entre contextos e entre atributos de contextos, respectivamente.

Figura 40: Implementação das funções de similaridade



Fonte: Elaborado pelo Autor

5.2 Configuração do processo de análise de similaridade

Conforme descrito na especificação do *framework*, detalhada no Capítulo 4, um processo de análise de similaridade é composto por um conjunto de tarefas (*tasks*), onde cada tarefa é implementada por uma classe, que por sua vez pode encapsular uma função de análise de similaridade, um filtro ou uma transformação nos dados. A classe *SIMCOPProcess* é responsável por encapsular a lista de tarefas e também por instanciar cada classe com seus respectivos parâmetros operacionais. Uma instância da classe *SIMCOPProcess* pode ser populada diretamente pela aplicação cliente, via código, ou configurada a partir de um arquivo de configuração. A Listagem 5 ilustra um arquivo de configuração que especifica o processo de análise de similaridade:

1. Uma etapa de **pré processamento** composta duas tarefas: alinhamento de sequências baseado no horário de cada contexto, e filtro por gaps, eliminando contextos que não possuem correpondência nas duas sequências, em função do horário;
2. A **análise de similaridade**, utilizando a função ECDefault para comparar cada par de contexto entre as sequências, retornando aqueles que obtiverem o valor mínimo de similaridade. Esta função usará a soma de similaridade entre atributos para determinar a similaridade entre dois contextos. Por sua vez, serão avaliados os atributos: Tempo (*TD*), através da função *TimeDifference*, e a *Situation (SD)* Temperatura, através da função *SituationNumericDifference* que calcula diferença entre atributos numéricos;
3. Uma etapa de **pós processamento**, composta por uma rotina de ordenação dos pares de contextos encontrados, baseada no valor de similaridade calculado para cada par.

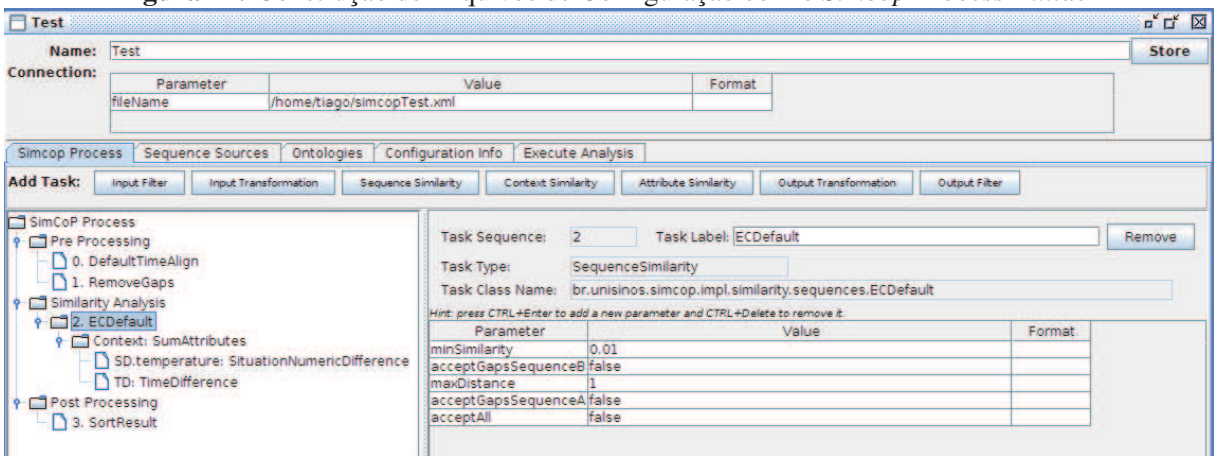
Listagem 5: Exemplo de configuração de um processo de análise de similaridade

```
<?xml version="1.0" encoding="UTF-8"?>
<SIMCOPConfig xmlns="http://www.unisinos.br/pipca/SIMCOP">
  <name>Test</name>
  <SIMCOPProcess>
    <!-- PRE-PROCESSAMENTO -->
    <task class="br.unisinos.SIMCOP.impl.transformations.DefaultTimeAlign" type="InputTransformation" />
    <task class="br.unisinos.SIMCOP.impl.filters.RemoveGaps" type="InputFilter" />
    <!-- ANALISE DE SIMILARIDADE: Entre Sequencias -->
    <task class="br.unisinos.SIMCOP.impl.similarity.sequences.ECDefault" type="SequenceSimilarity" >
        <parameter name="acceptAll" value="false"/>
        <parameter name="acceptGapsSequenceA" value="false"/>
        <parameter name="acceptGapsSequenceB" value="false"/>
        <parameter name="maxDistance" value="1"/>
        <parameter name="minSimilarity" value="0.01"/>
        <!-- ANALISE DE SIMILARIDADE: Entre Contextos -->
        <task class="br.unisinos.SIMCOP.impl.similarity.contexts.SumAttributes" type="ContextSimilarity">
           <parameter name="DistanceOrSimilarity" value="both"/>
           <!-- ANALISE DE SIMILARIDADE: Entre Atributos -->
           <task for="SD.temperature" type="AttributeSimilarity"
                 class="br.unisinos.SIMCOP.impl.similarity.attributes.SituationNumericDifference" >
              <parameter name="absoluteDifference" value="true"/>
              <parameter name="missingValue" value="0"/>
           </task>
           <task for="TD" type="AttributeSimilarity"
                 class="br.unisinos.SIMCOP.impl.similarity.attributes.TimeDifference">
              <parameter name="scale" value="minutes"/>
              <parameter name="missingValue" value="0"/>
              <parameter name="absoluteDifference" value="true"/>
           </task>
```

```
30          </task>
31      </task>
32      <!-- POS-PROCESSAMENTO -->
33      <task class="br.unisinos.SIMCOP.impl.transformations.SortResult" type="OutputTransformation" >
34          <parameter name="sortBy" value="calculatedValue"/>
35      </task>
36    </SIMCOPProcess>
37  </SIMCOPConfig>
```

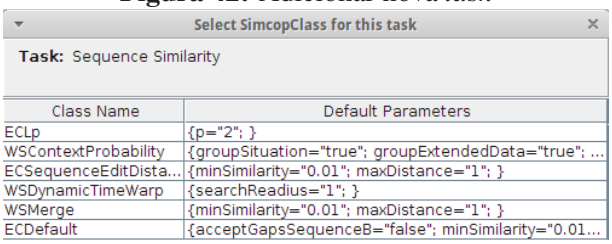
Para facilitar a construção de arquivos de configuração, foi desenvolvida uma ferramenta chamada *SIMCOPProcessBuilder*. A Figura 41 mostra o uso deste aplicativo para edição do arquivo de configuração da Listagem 5, denominado "*Test*". O componente de configuração do *framework* permite que a configuração seja armazenada em qualquer meio de armazenagem, denominado *ConfigStore*. Neste exemplo, está se utilizando um *ConfigStore* que armazena a configuração em um arquivo XML apontado pelo parâmetro de conexão "*fileName*". O processo de análise de similaridade está representado na parte inferior esquerda da Figura 41, onde as tarefas estão agrupadas conforme a etapa de processamento: *Pre Processing*, *Similarity Analysis* e *Post Processing*. Novas tarefas podem ser adicionadas utilizando-se os botões "*Add Task*", conforme ilustrado na Figura 42. Por fim, os parâmetros operacionais de cada tarefa podem ser configurados na parte inferior direita da tela.

Figura 41: Construção de Arquivos de Configuração com o *Simcop Process Builder*



Fonte: Elaborado pelo Autor

Figura 42: Adicionar nova *task*

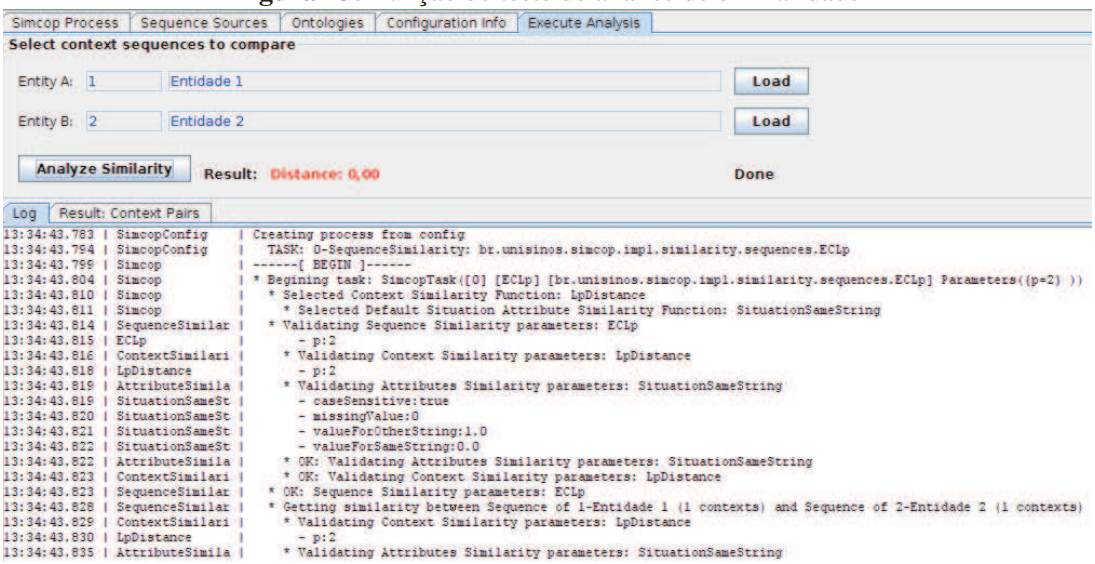


Fonte: Elaborado pelo Autor

Por fim, a ferramenta permite que sejam realizados testes de análise de similaridade utilizando a configuração construída. Esta funcionalidade está disponibilizada através da aba "*Exe-*

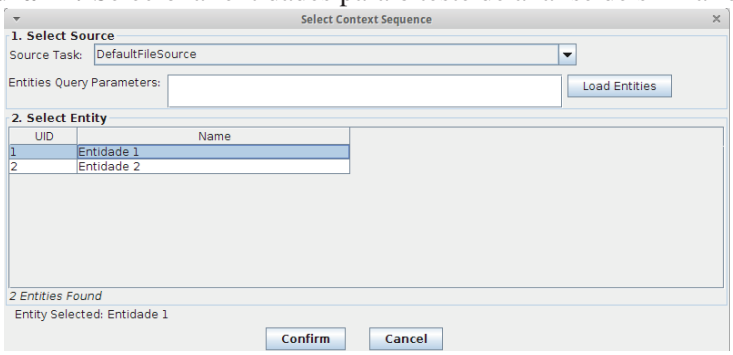
cute Analysis", conforme a Figura 43. Inicialmente seleciona-se duas entidades, a partir dos *SequenceSource*'s configurados conforme a Figura 44 e em seguida clica-se no botão "*Analyze Similarity*". Será exibido o valor de similaridade entre as sequências de contextos das duas entidades, bem como o *log* de processamento. Também são exibidos os pares de contextos retornados no resultado e o valor de similaridade ou distância de cada par, conforme a Figura 45.

Figura 43: Função de teste de análise de similaridade



Fonte: Elaborado pelo Autor

Figura 44: Selecionar entidades para o teste de análise de similaridade



Fonte: Elaborado pelo Autor

Os arquivos de configuração gerados pela ferramenta *SIMCOPProcessBuilder* definem processos de análise de similaridade que são independentes e portanto ser reaproveitados em qualquer aplicação que esteja utilizando o *Framework SIMCOP*.

5.3 Avaliação do SIMCOP

Através da implementação do protótipo foi avaliada a capacidade do *framework* de identificar corretamente as similaridades entre contextos em situações reais. Com este objetivo

Figura 45: Resultado do teste de análise de similaridade

In...	Context A	Context B	Calculated Va...
00	Time: 2013-12-23 00:00:00.0 Location: UNDEFINED creator1: ESCOSSIA LILIANA DA creator2: KASTRUP VIRGINIA creator3: PASSOS EDUARDO date: 2009 format: IMPRESSO language: POR publisher: SULINA subject1: PSICANALIS subject2: CARTOGRAF subject3: PSICOLOG title: PISTAS DO METODO DA CARTOGRAFIA PESQUISA INTERVENC type: LIVRO	Time: 2013-12-22 00:00:00.0 Location: UNDEFINED creator1: ESCOSSIA LILIANA DA creator2: KASTRUP VIRGINIA creator3: PASSOS EDUARDO date: 2009 format: IMPRESSO language: POR publisher: SULINA subject1: PSICANALIS subject2: CARTOGRAF subject3: PSICOLOG title: PISTAS DO METODO DA CARTOGRAFIA PESQUISA INTERVENC type: LIVRO	0,00

Fonte: Elaborado pelo Autor

foram implementados dois sistemas de recomendação (AL-KHALIFA, 2008; ANAND; BHARADWAJ, 2011; HONG; LI; LI, 2012) e avaliados em função da qualidade das recomendações oferecidas aos usuários. De maneira geral, estes sistemas dependem, de forma explícita ou implícita, da análise de similaridade para encontrar itens que possuam alguma semelhança com itens de interesse do usuário. A proposta destas avaliações é a representação do histórico de itens acessados por usuários como sequências de contextos, permitindo a avaliação de semelhanças entre históricos visando a recomendação de itens.

Com estas aplicações também foi possível avaliar a viabilidade da representação de conjuntos de dados concretos através do modelo genérico de sequências de contextos proposto pelo *framework*. Em ambas as aplicações, os usuários correspondem as entidades (*Entity*) e o histórico de itens acessados por cada usuário correspondem as sequencias de contextos (*ContextSequence*) visitados por estas entidades. Verificou-se que qualquer objeto, vinculado a uma entidade e possível de ser descrito como um conjunto de situações em algum momento, e opcionalmente em algum lugar, pode ser tratado como um contexto (*Context*) desta entidade, conforme a definição de DEY; ABOWD; SALBER (2001). Cada situação destes contextos, por sua vez, pode ser descrita na forma *"predicado"="valor"*, conforme o modelo de HECKMANN (2005), de onde conclui-se que qualquer registro que descreva uma ou mais situações de uma entidade, em algum momento possa ser interpretado como um contexto da entidade.

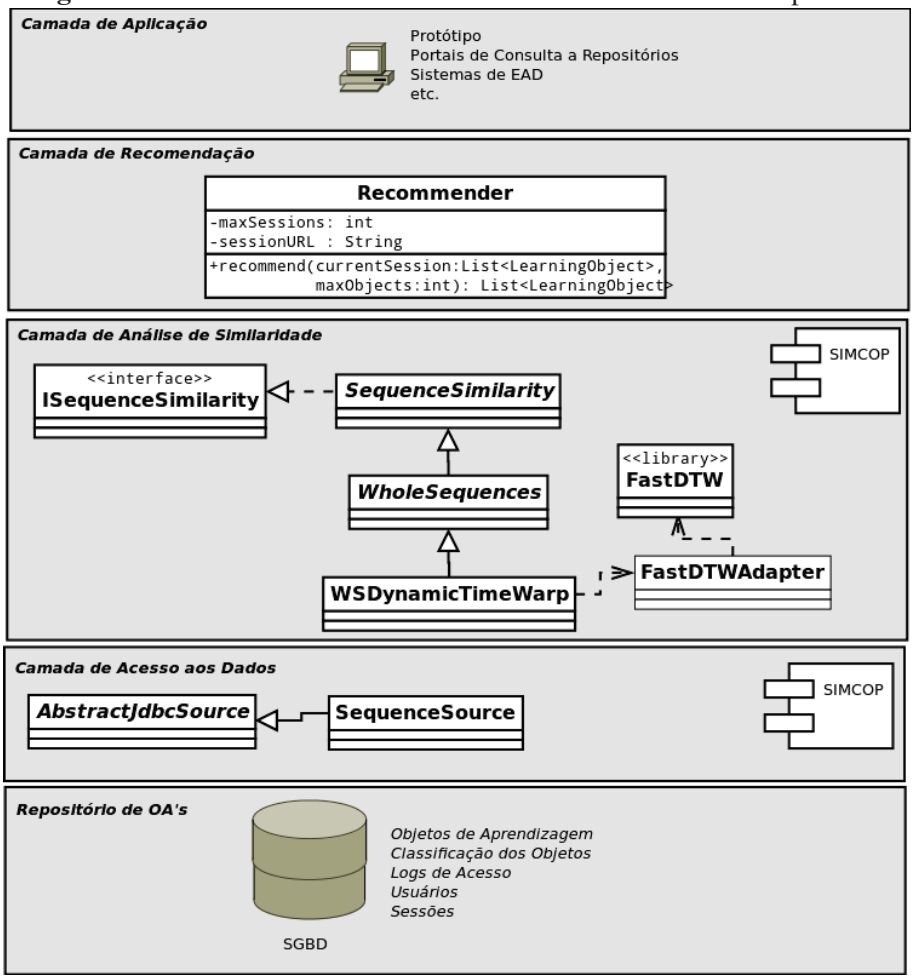
5.3.1 ReBaSS: Um modelo de recomendação de objetos de aprendizagem baseado em similaridade

O ReBaSS - Recomendação Baseada em Similaridade de Sessões (WIEDEMANN; BARBOSA; RIGO, 2013), é um modelo para recomendação de Objetos de Aprendizagem (OA) que leva em consideração os objetos acessados durante uma sessão. Considera-se "sessão" as atividades executadas por um usuário em um ambiente de consulta de objetos de aprendizagem após o seu login. O modelo proposto recebe a sequência de OA consultados durante a sessão atual do usuário e localiza sessões cujas sequências de OA consultados sejam similares à sequência da sessão corrente. Os OA encontrados nas sessões similares são então recomendados ao usuário.

A Figura 46 mostra a arquitetura do modelo. A camada de aplicação refere-se a interface gráfica utilizada para acesso ao repositório de OA's na qual será implementada a função de

recomendação proposta pelo modelo. A camada de recomendação é responsável por identificar, nas sessões similares, OA's que ainda não tenham sido consultados na sessão atual. Esta camada utiliza a camada de Análise de Similaridade para identificar, no histórico de sessões do repositório, sessões cuja lista de OA's seja semelhante a lista de OA's da sessão atual. O SIMCOP é usado nesta camada, para realizar o processo de análise de similaridade, e também na camada seguinte, chamada de camada de acesso aos dados, e responsável por acessar o histórico de sessões no repositório e mapear os dados obtidos para o SIMCOP. Por fim, a camada denominada repositório corresponde ao meio de armazenamento físico dos OA's, normalmente em um banco de dados.

Figura 46: Modelo do ReBaSS utilizando o SIMCOP como componente



Fonte: Elaborado pelo Autor

O protótipo do SIMCOP foi utilizado como um componente da camada de análise de similaridade, com a tarefa de identificar semelhanças entre as listas que compõem as sessões armazenadas nos históricos de consulta do repositório de OA's. Para este fim, foi necessário implementar, na camada de acesso aos dados, uma classe chamada *SequenceSource* responsável por realizar o mapeamento do modelo de dados da aplicação ReBaSS, composto por listas de OA's, para o SIMCOP, composto por sequencias de contextos, cada sessão foi tratada como

sendo uma Entidade e a lista de OA's consultados em cada sessão como sendo a Sequência de Contextos desta entidade. Os atributos de cada OA consultado, como nome, código do curso e módulo foram mapeados como sendo as Situações (*Situation*) de cada contexto. A implementação da classe *SequenceSource* é exibida na Listagem 6.

Listagem 6: *SequenceSource* implementado para o protótipo do ReBaSS

```
/**
 * Transforma uma lista de objetos de aprendizagem (OA) acessados durante uma sessao em uma sequencia de
 * contextos visitados por uma entidade (usuario). Os dados do OA sao tratados como atributos do contexto.
 */
public class SequenceSource extends AbstractJdbcSource {
    private PreparedStatement stmSequence;
    /** Acesso ao banco de dados */
    public boolean connectToSource() {
        try {
            Class.forName("org.postgresql.Driver");
            this.connection = DriverManager.getConnection("jdbc:postgresql://localhost/rebass", "postgres", "postgres");
            return !this.connection.isClosed();
        } catch (Exception e) {
            Logger.getLogger(SequenceSource.class.getName()).log(Level.SEVERE, null, e);
            return false;
        }
    }
    /* Obter a lista de sessoes mapeadas no banco de dados */
    public List<Entity> getListEntities(Object... queryParameters) throws Exception {
        ResultSet rs = connection.prepareStatement("select distinct id from sessao").executeQuery();
        List<Entity> result = new ArrayList<Entity>();
        while (rs.next()) {
            Long sessionid = rs.getLong("id");
            result.add( loadEntity(sessionid) );
        }
        return result;
    }
    /** Transformar sessao em entidade */
    public Entity loadEntity(Object uid) throws Exception {
        return new Entity(uid, "Sessao " + uid);
    }
    /**
     * Criar uma entidade e sua sequencia de contextos a partir dos OA's selecionados durante a sessao atual do
     * usuario. Este metodo trabalha sobre a lista de objetos em memoria e nao acessa o SGBD.
     */
    public ContextSequence carregarSequenciaAtual(List<MdlLog> lstAtual, String usuario) {
        Entity entity = new Entity(0, usuario);
        ContextSequence cs = new ContextSequence(entity);
        for (MdlLog mdl : lstAtual) {
            Context ctx = createContext(mdl.getDatahora(), mdl.getIp(), mdl.getCourse().longValue(),
                                        mdl.getCmid().longValue(), Long.parseLong(mdl.getInfo()),
                                        mdl.getAction(), mdl.getResourceName(), mdl.getResourceIntro());
            cs.addContext(ctx);
        }
        return cs;
    }
    /**
     * Carregar a lista de OA's de uma sessao armazenada no SGBD transformando-a em uma sequencia de contextos.
     */
    public void loadIntoSequence(ContextSequence sequence) throws Exception {
        if (stmSequence == null) {
            StringBuilder sql = new StringBuilder();
            sql.append("select l.datahora, l.ip, l.course, l.cmid, l.resource_id, l.action, r.name, r.intro ");
            sql.append("  from mdl_log l");
            sql.append("       inner join mdl_resource r on l.course = r.course and l.resource_id = r.id");
            sql.append(" where module='resource' and sessionid = ?");
            sql.append("   order by datahora");
            stmSequence = this.connection.prepareStatement(sql.toString());
        }
        stmSequence.setLong(1, (Long) sequence.getEntity().getUid());
        ResultSet rs = stmSequence.executeQuery();
        while (rs.next()) {
            Date time       = rs.getTimestamp("datahora");
            String ip       = rs.getString("ip");
            Long course     = rs.getLong("course");
            Long cmid       = rs.getLong("cmid");
            Long resourceID = rs.getLong("resource_id");
            String action   = rs.getString("action");
            String name     = rs.getString("name");
            String intro    = rs.getString("intro");
```

```
71              Context ctx = createContext(time, ip, course, cmid, resourceID, action, name, intro);
72              sequence.addContext(ctx);
73          }
74      }
75      /**
76       * Converte um OA em um contexto, considerando cada atributo do OA como uma situacao (Situation) do contexto.
77       * @param time Horario de acesso ao OA (Time).
78       * @param ip Origem do acesso ao OA, pode ser tratado como localizacao (Location) do contexto.
79       * @param course Identificacao do curso no qual o OA foi publicado (Situation).
80       * @param cmid Identificacao do modulo do curso no qual o OA foi publicado (Situation).
81       * @param resourceid Identificacao do OA (Situation).
82       * @param action Acao realizada no ambiente Moodle para acesso a este resource (Situation).
83       * @param name Titulo do OA (Situation).
84       * @param intro Texto opcional descritvo do OA (Situation).
85       * @return OA Transformado em Contexto visitado
86       */
87      private Context createContext(Date time, String ip, Long course, Long cmid, Long resourceid,
88                                    String action, String name, String intro) {
89              Context ctx = new Context();
90              ctx.setTime(new TimeDescription(time));
91              ctx.setLocation(new LocationDescription(ip));
92              ctx.addSituation("course"    , SIMCOPValue.createFromObject(course));
93              ctx.addSituation("cmid"      , SIMCOPValue.createFromObject(cmid));
94              ctx.addSituation("resourceid", SIMCOPValue.createFromObject(resourceid));
95              ctx.addSituation("action"    , SIMCOPValue.createFromObject(action));
96              ctx.addSituation("name"      , SIMCOPValue.createFromObject(name));
97              ctx.addSituation("intro"     , SIMCOPValue.createFromObject(intro));
98              return ctx;
99      }
100     /** Obrigatorio para a interface. Nenhum parametro adicional necessario nesta implementacao. */
101     public Parameters getDefaultParameters() { return null; }
102 }
```

O processo de análise de similaridade usado no desenvolvimento do ReBaSS e executado pelo SIMCOP utiliza a técnica conhecida como DTW (Dynamic Time Warping), descrita em detalhes na seção 2.7.2. Esta técnica trata as sequências de contextos como séries temporais, onde cada série corresponde a uma sequencia, cada ponto das séries corresponde a um contexto e o valor para comparação usado pela técnica consiste no resultado de uma função de distância que realiza a comparação entre cada contexto alinhado. Para implementação da técnica DTW foi utilizada uma biblioteca *Open Source* chamada *FastDTW*[1], que é utilizada pelo SIMCOP através de uma classe denominada *FastDTWAdapter*, que converte o modelo de dados do SIMCOP para o modelo de dados da biblioteca.

Para validação deste modelo foi implementado um protótipo que compara a lista de OA's consultados durante a sessão atual de um usuário no ambiente EAD *Moodle*[2] de uma universidade[3], com a lista de OA's em sessões previamente armazenadas no banco de dados. As sessões cuja lista de OA's forem similares a lista da sessão atual são consideradas sessões similares. Pressupõe-se que os interesses dos usuários das sessões similares também sejam similares aos interesses atuais do usuário, tendo em vista a semelhança entre as listas de OA consultados. Partindo deste pressuposto, o protótipo localiza nas sessões similares, OA's que ainda não foram consultados durante a sessão atual.

Uma visão geral da tela de consulta desenvolvida para o protótipo pode ser vista na Figura 47. Na parte esquerda da tela, o usuário localiza OA's que são do seu interesse de pesquisa atual. Neste protótipo, conforme a Figura 48, foi criada a opção "Selecionar" que simula a

[1] http://code.google.com/p/fastdtw/
[2] http://moodle.org
[3] http://fit.faccat.br

operação de acesso a OA's normalmente executada em sistemas de EAD ou repositórios *on-line* de OA. Por fim, a lista de recomendações é construída a partir de OA's encontrados nas sessões similares, identificadas através da análise de similaridade entre estas e a lista de OA's selecionados na sessão atual.

Figura 47: Tela do protótipo do ReBaSS

Fonte: Elaborado pelo Autor

As Figuras 48 e 49 ilustram a eficácia obtida na recomendação baseada em similaridade. A primeira ilustra uma sessão de usuário que está acessando OA's referentes a textos do Prof. Andrew Tanenbaum sobre Sistemas Distribuídos, que corresponde ao interesse de pesquisa atual do aluno. A segunda figura ilustra os OA's recomendados, através da análise de similaridade entre a sessão atual e sessões antigas armazenadas no banco de dados do *Moodle*. Objetos presentes em sessões similares que ainda não tenham sido acessados durante a sessão atual são então recomendados ao usuário. Nesta lista de recomendações é possível perceber que o sistema recomendou OA's que mencionam o Prof. Tanenbaum, porém foram recomendados dois objetos ("Petru Eles: apresentação resumo" e "Prova G2 2011/1") na qual não existe esta menção. Isto demonstra a capacidade do protótipo de recomendar OA's de interesse do aluno, mesmo que não haja um vínculo explícito entre estes OA's no repositório. A possibilidade dos OA's recomendados serem de interesse do aluno é inferida a partir da semelhança entre a lista de OA's da sessão atual e as listas de OA's das sessões das quais estes OA's foram obtidos.

Figura 48: ReBaSS: Selecionar OA

Fonte: Elaborado pelo Autor

Os resultados da pesquisa com o ReBaSS foram publicados no 24º Simpósio Brasileiro de Informática na Educação (SBIE) realizado na cidade de Campinas/SP em Novembro/2013. O

Figura 49: ReBaSS: Recomendação

Fonte: Elaborado pelo Autor

artigo publicado foi classificado para a sessão de *Best Papers* e obteve a 2º colocação, tendo sido convidado para publicação de uma versão estendida na Revista Brasileira de Informática na Educação (RBIE).

Através deste experimento, foi possível concluir que o SIMCOP é capaz de analisar e identificar corretamente as similaridades entre sequencias de contextos. O *framework* também mostrou-se capaz de tratar as estruturas de dados específicas do domínio da aplicação, no caso Objetos de Aprendizagem, através do modelo genérico de Sequências de Contextos. O fato do componente de análise de similaridade do ReBaSS ter sido capaz de identificar semelhanças entre listas de objetos que não possuíam vínculos entre si, demonstra a eficácia da proposta do SIMCOP em abstrair dados da aplicação na forma de sequências de contextos para então aplicar técnicas de análise de similaridade já utilizadas em outros domínios, como é o caso da *DTW*.

5.3.2 Um componente de Filtragem Colaborativa para o modelo *U-Library*

O protótipo foi utilizado por um desenvolvedor independente, na implementação do componente de recomendação de materiais bibliográficos do *U-Library*, um modelo de biblioteca ubíqua implementado para avaliação na UNIVATES[4]. Neste cenário, a função do SIMCOP foi a de realizar análises de similaridade visando encontrar usuários cuja sequência de livros retirados seja similar a do usuário para a qual a recomendação será feita. Nesta seção são descritos os procedimentos adotados por este desenvolvedor para a implementação do sistema, os resultados obtidos e as críticas quanto ao uso do *framework*.

Inicialmente, foi implementado o mapeamento dos dados disponíveis à aplicação de recomendação para o modelo de dados do SIMCOP, através da implementação de uma classe *SequenceSource*. Cada usuário foi tratado como uma entidade, e cada livro retirado foi tratado como um contexto, sendo que os atributos de cada livro foram tratados como situações (*Situation)* de cada contexto. A Tabela 9 ilustra os dados de retiradas de livros disponíveis à aplicação, com seus respectivos atributos.

[4]http://www.univates.br

Tabela 9: Exemplo de atributos disponíveis nos contextos do sistema de recomendação

Metadados	Valor
Time	2013-12-24
Location	UNDEFINED
creator	ESCOSSIA LILIANA DA
creator	KASTRUP VIRGINIA
creator	PASSOS EDUARDO
date	2009
format	IMPRESSO
language	POR
publisher	SULINA
subject	PSICANALIS
subject	CARTOGRAF
subject	PSICOLOG
title	PISTAS DO METODO DA CARTOGRAFIA PESQUISA INTERVENCAO E PRODUCAO DE SUBJETIVIDADE
type	LIVRO

Fonte: Conforme informado pelo desenvolvedor

Após realizado o mapeamento destes dados para contextos do SIMCOP, foram aplicados os seguintes critérios para análise de similaridade:

- Todos usuários foram mapeados como entidades;
- Cada livro retirado pelos usuários foi mapeado como um contexto;
- Os metadados dos livros foram mapeados como Situations para um contexto;
- Foi realizada a análise de similaridade conforme illustrado na Listagem 7;
- As recomendações foram restritas a usuários que atingirem no mínimo 0.2 de similaridade, considerando suas sequências de livros retirados (contextos);
- Somente foram considerados pares de contextos que atingiram no mínimo 0.5 de similaridade.

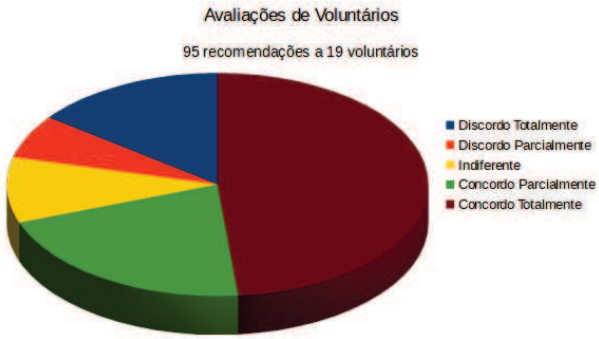
Listagem 7: Processo de análise de similaridade utilizado para Filtragem Colaborativa

```
1  SequenceSimilarity sequenceSimilarity = new ECDefault();
2  sequenceSimilarity.setParameter(ECDefault.ACCEPT_ALL, "false");
3  sequenceSimilarity.setParameter(ECDefault.ACCEPT_GAPS_SEQUENCE_A, "false");
4  sequenceSimilarity.setParameter(ECDefault.ACCEPT_GAPS_SEQUENCE_B, "false");
5  sequenceSimilarity.setParameter(ECDefault.MIN_SIMILARITY, "0.01");
6
7  ContextSimilarity ctxSimilarity = new SimpleAttributes();
8  ctxSimilarity.setParameter(SimpleAttributes.COMPARE_TIME, "false");
9  ctxSimilarity.setParameter(SimpleAttributes.COMPARE_LOCATION, "false");
10 ctxSimilarity.setParameter(SimpleAttributes.COMPARE_SITUATIONS, "true");
11 ctxSimilarity.setParameter(SimpleAttributes.COMPARE_EXTENDED_DATA, "false");
```

Após a implementação do componente de recomendação, foi realizada uma avaliação com dezenove voluntários, que classificaram as recomendações recebidas em cinco categorias. O gráfico da Figura 50 mostra o resultado das avaliações: quatorze (14,7%) recomendações foram avaliadas como “Discordo Totalmente”, seis (6,3%) foram avaliadas como “Discordo Parcialmente”, nove (9,5%) foram avaliadas como “Indiferente”, vinte (21,1%) foram avaliadas como

"Concordo Parcialmente" e quarenta e seis (48,4%) foram avaliadas como "Concordo Totalmente", perfazendo um total de noventa e cinco recomendações avaliadas.

Figura 50: Avaliações da Filtragem Colaborativa utilizando o SIMCOP



Fonte: Elaborado pelo Autor

Por fim, foi solicitado ao desenvolvedor que respondesse a um breve questionário, descrevendo sua experiência no uso do SIMCOP. A seguir são transcritas as perguntas e as respectivas respostas do desenvolvedor.

1. **O protótipo testado atendeu as necessidades da aplicação?** Sim, o protótipo permitiu a implementação de um motor de recomendações de materiais bibliográficos para o modelo U-Library, a partir da análise de similaridade de trilhas de bibliotecas.

2. **O protótipo foi capaz de identificar corretamente similaridades nos dados de teste?** Sim, através do protótipo foi possível testar diversos algoritmos de similaridades de trilhas, permitindo a identificação do algoritmo mais adequado aos objetivos do experimento.

3. **O tempo de resposta das análises de similaridade foi adequado?** Sim, o tempo de resposta foi adequado mesmo para uma grande quantidade de dados.

4. **O código fonte disponível estava claro?** Sim, o código fonte é claro e bem comentado.

5. **Na sua opinião, o que poderia ser melhorado na implementação ou especificação do *framework*?** Os experimentos realizados utilizando o SIMCOP tiveram o objetivo de permitir a identificação e entrega de recomendações de materiais bibliográficos do modelo U-Library. Desta forma, os dados utilizados para os experimentos são essencialmente textuais, de modo que encontrou-se dificuldades em tratar o alinhamento de metadados. Outro ponto que creio que poderia ser considerado, é a possibilidade de efetuar tratamentos em conteúdos textuais, como a extração de radicais de palavras e ignorar stopwords. No entanto, estas questões também podem ser tratadas pelo projeto utilizador do framework SIMCOP, como ocorreu com o U-Library.

A Tabela 10 ilustra o problema descrito pelo desenvolvedor na pergunta 5 como "alinhamento de metadados". Dois livros (*Context*) tratam exatamente dos mesmos assuntos e cada

assunto corresponde a uma *Situation*, que neste exemplo são identificadas pelos predicados: "*subject1*", "*subject2*" e "*subject3*". Porém, devido a ordem na qual os dados são obtidos pela aplicação cada *subject* recebe um valor diferente, o que faz com que estes contextos sejam considerados não-similares, ainda que o conjunto de *subject*'s seja igual nos dois contextos.

Tabela 10: Exemplo de desalinhamento de *Situations*

Predicado	Entidade 1	Entidade 2
subject1	MEDICINA	INFANTIL
subject2	ENFERMAGEM	MEDICINA
subject3	INFANTIL	ENFERMAGEM

Fonte: Conforme informado pelo desenvolvedor

A especificação do *framework* não prevê suporte explícito a este tipo de situação, uma vez que cada predicado deve ser independente de outro. Esperava-se que esta situação fosse tratada através do *SequenceSource* no momento da transformação dos dados da aplicação em contextos. Entretanto este tratamento pode se tornar uma tarefa complexa, dependendo da estrutura dos dados disponibilizados pela aplicação. Portanto, este é um ponto que deverá ser aperfeiçoado em versões futuras do *framework*.

A experiência do uso do SIMCOP por um desenvolvedor independente permitiu avaliar aspectos relativos a qualidade do código do protótipo implementado e também a clareza do *framework* especificado. Com ajuda de exemplos de uso, acompanhados da documentação do *framework*, o desenvolvedor foi capaz de utilizar o *framework* como um componente de sua aplicação, implementando a classe *SequenceSource* para realizar o mapeamento dos dados da aplicação para o modelo de sequencias de contextos, assim como especificar um processo de análise de similaridade que atendesse as seus requisitos. O processo de análise de similaridade especificado pelo desenvolvedor poderá inclusive ser reaproveitado em outras aplicações, através do uso do componente de configuração disponbilizado pelo SIMCOP.

A qualidade das análises de similaridade pôde ser avaliada por um grupo de voluntários, que avaliaram as recomendações geradas a partir da análise de similaridade. Neste experimento, 69,5% das recomendações oferecidas foram aceitas pelos usuários. Apesar deste numero poder ser considerado satisfatório para a primeira versão do protótipo, já que a maioria das recomendações foi aceita, conclui-se que a implementação das técnicas de análise de similaridade ainda pode ser aperfeiçoada, tendo em vista as limitações identificadas pelo desenvolvedor, referentes a análise de dados textuais e alinhamento de situações.

5.4 Considerações sobre o capítulo

Neste capítulo foram apresentados os detalhes de implementação do protótipo do SIMCOP, bem como descritos os resultados alcançados em dois estudos de casos. Foi exemplificada a implementação do *hot spot SequenceSource*, responsável por realizar o mapeamento dos dados de uma aplicação para o modelo de dados do SIMCOP. Também foi ilustrada a utilização do

SIMCOP para a realização da análise de similaridade e apresentada a ferramenta *SIMCOPProcessBuilder* disponível para a criação de arquivos de configuração, permitindo que processos de análise de similaridade sejam compartilhados entre aplicações. Por fim, foram apresentados os resultados obtidos na utilização do protótipo do *framework* em dois sistemas de recomendações reais e descritas as avaliações obtidas dos usuários e da eficácia do processo de análise de similaridade.

6 CONSIDERAÇÕES FINAIS

Esta dissertação abordou o problema da análise de similaridade em sequências de contextos de entidades genéricas. Foi desenvolvida a especificação de um *framework*, chamado *SIMCOP* (SIMilar COntext Path), que permite a utilização combinada de diferentes técnicas de análise de similaridade para se calcular a similaridade global entre duas sequências.

O *SIMCOP* oferece um conjunto de classes que implementam diversas funções de análise de similaridade. Entretanto o utilizador do *framework* poderá desenvolver novas classes de análise implementando as *interfaces* ou extendendo uma das classes disponíveis no *framework*.

A configurabilidade do *SIMCOP*, descrita no capítulo 5, permite definir processos de análise de similaridade independentes de aplicação, permitindo seu reaproveitamento bem como a utilização e comparação de diferentes processos em uma mesma aplicação. O componente de configuração permite determinar quais as classes de cálculo de similaridade serão utilizadas para cada categoria de dado contextual presente nas sequências analisadas, bem como qual a classe que implementará a análise global de similaridade, combinando os resultados das classes de cálculo. Também poderão ser configurados os parâmetros operacionais de cada classe e definidos os procedimentos de pré e pós processamento, como aplicação de filtros, limpeza e sumarização de dados e a aplicação de técnicas de *KDD*.

Este capítulo está dividido em três seções. A primeira descreve as conclusões gerais da pesquisa realizada. A segunda descreve as contribuições do *framework* em comparação com os trabalhos relacionados. Na terceira seção são descritos os trabalhos futuros.

6.1 Conclusões

O *framework* SIMCOP, especificado no Capítulo 4, mostrou-se capaz de realizar analises de similaridade em situações de aplicações reais e obtendo resultados satisfatórios conforme avaliações demonstradas no Capítulo 5. Entretanto ainda há a necessidade de melhorias em alguns aspectos do *framework*, especialmente no que se refere ao tratamento de dados textuais, conforme apontado por um desenvolvedor independente, que utilizou o SIMCOP na implementação de um componente de recomendação em um modelo de biblioteca Ubiqua.

A utilização do SIMCOP por um desenvolvedor independente permitiu uma avaliação subjetiva da qualidade da análise de similaridade, feita por usuários reais. O componente implementado com o uso do protótipo do SIMCOP implementou um processo de análise de similaridade para comparação de históricos de consultas de livros, visando a identificação de interesses similares para a realização de recomendações de livros usando a técnica de filtragem colaborativa. Foi realizado um experimento, no qual 95 recomendações foram apresentadas a 19 usuários, que avaliaram a relevância das recomendações, classificando-as em 5 critérios: “Discordo totalmente”, “Discordo parcialmente”, “Indiferente”, “Concordo parcialmente” e “Concordo totalmente”. Neste experimento os usuários concordaram com 69,5% das recomendações, dis-

cordaram de 21% e consideraram indiferentes 9,5%. Considerando-se os apontamentos feitos pelo desenvolvedor, especialmente na questão do alinhamento de situações e tratamento de dados textuais, poderá ser possível aperfeiçoar a qualidade da análise de similaridade, melhorando desta forma a relevância das recomendações efetuadas auxiliadas pelo SIMCOP.

Em um segundo experimento, o SIMCOP foi utilizado para a implementação do protótipo de um modelo de recomendação de objetos de aprendizagem (OA) que propõe o uso da análise de similaridade entre históricos de sessões de consultas em repositórios, para identificação de OA's que possam estar relacionados aos interesses atuais do aluno. No experimento realizado, utilizando-se uma base de dados real mantida pelo sistema de EAD de uma universidade, o protótipo desenvolvido para testar este modelo foi capaz de identificar objetos possivelmente relacionados aos interesses do aluno, mesmo quando não havia vinculo, explícito ou implícito, entre os objetos consultados durante a sessão atual com os objetos recomendados. Neste cenário, o SIMCOP foi capaz de identificar corretamente as similaridades entre os históricos de consultas a OA's, usando um processo de análise de similaridade baseado na técnica DTW (*Dynamic Time Warping*) que trata os históricos como séries temporais de contextos visitados.

Por fim, foi utilizado o modelo de HECKMANN (2005) para representação de contextos, e a partir dele criado o modelo de dados do SIMCOP. Este modelo mostrou ser capaz de representar satisfatoriamente diversos tipos de informações como sendo contextos de uma entidade. Concluiu-se que qualquer sequência de registros que esteja vinculada a uma entidade, indexado cronologicamente e que descreva situações que possam ser representadas no formato *"predicado"="valor"*, podem ser representadas como sequências de contextos, permitindo a analise de similaridade através das ferramentas disponibilizadas pelo *framework*.

6.2 Contribuições

A Tabela 11 compara as características do SIMCOP com os trabalhos relacionados estudados no Capítulo 3. A principal contribuição deste trabalho foi a criação de uma arquitetura extensível e configurável que permite combinar diferentes técnicas de análise de similaridade para identificar semelhanças entre duas sequências de contextos de entidades genéricas. Apesar do *SIMCOP* permitir que sejam implementadas métricas de análise de similaridade que utilizem ontologias, não foi implementado nesta versão do *framework* nenhum mecanismo explícito para leitura e interpretação de arquivos OWL ou RDF. Portanto esta contribuição será introduzida em uma versão futura do *framework*.

O modelo de dados utilizado pelo *framework* permite abstrair diferentes tipos de dados na forma de sequências de contextos. Somando-se a isto a capacidade de combinar diferentes técnicas de análise de similaridade em um componente único, torna-se possível a aplicação destas técnicas em diferentes cenários, bastando apenas que os desenvolvedores realizem o mapeamento dos dados disponíveis em suas aplicações para o modelo do SIMCOP, através da implementação de uma classe *SequenceSource*. Segundo um desenvolvedor independente,

Tabela 11: Contribuições em relação aos trabalhos relacionados

Características		SmartTrace* ZEINALIPOUR-YAZTI et al. (2013)	Group Discovery Framework LI et al. (2012)	TraSimilar ABRAHAM; LAL (2012)	WebTraSim ABRAHAM; LAL (2011)	Routine Activity LV; CHEN; CHEN (2013)	TSC CUI; ZHAO; TOK (2012)	SeqSim OBWEGER et al. (2010)	SimCoP
Categoria de dados contextuais DEY; ABOWD; SALBER (2001)	**Entidade** *(Identity)*	Usuários de Smartphone	Genérica	Veículos	Usuários Web	Usuários	Genérica	Business Events	Genérica
	Tempo *(Hora, Data, ...)*	Independe	Data / Hora	Data / Hora	Data / Hora	Hora do dia	Independe	Data / Hora	Independe
	Posição Geográfica *(Location)*	**SIM**	**SIM**	**SIM**	*não*	**SIM**	**SIM**	*não*	**SIM**
	Situação *(Descrita por variáveis)*	*não*	*não*	*não*	*não*	*não*	**SIM**	**SIM**	**SIM**
	Atividade *(O que a entidade está fazendo)*	*não*	*não*	*não*	**SIM**	**SIM**	*não*	**SIM**	**SIM**
Tipo de Similaridade	**Valor numérico**	**SIM**	**SIM**	*não*	*não*	*não*	**SIM**	*não*	**SIM**
	Semântica	*não*	*não*	*não*	*não*	**SIM**	*não*	**SIM**	**SIM**
	Ontologias	*não*	*não*	*não*	*não*	*não*	*não*	*não*	**NÃO**
	Sintaxe	*não*	não	**SIM**	**SIM**	*não*	*não*	*não*	**SIM**
Forma de análise	**Dados Sumarizados**	não	não	não	não	**SIM**	**SIM**	não	**SIM**
	Item-a-Item	**SIM**	**SIM**	**SIM**	**SIM**	não	não	**SIM**	**SIM**
	Intersecção de Conjuntos	*não*	*não*	**SIM**	**SIM**	*não*	*não*	*não*	**SIM**
Recursos oferecidos	**Pré Processam.**	não	não	**SIM**	**SIM**	**SIM**	**SIM**	*não*	**SIM**
	Pós Processam.	*não*	*não*	*não*	*não*	*não*	*não*	*não*	**SIM**
	Multi-Métrica	*não*	*não*	*não*	*não*	*não*	*não*	*não*	**SIM**
	Métrica própria	**SIM**	*não*	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**	**SIM**
	Seleção de métrica	**SIM**	*não*	*não*	*não*	*não*	**SIM**	*não*	**SIM**
	Configurável	*não*	**SIM**	*não*	*não*	*não*	*não*	**SIM**	**SIM**

Fonte: Elaborado pelo autor

no experimento desenvolvido o *framework* mostrou ser de fácil compreensão e não apresenta grandes dificuldades à implementação.

Por fim, cabe ressaltar que além da utilização direta no desenvolvimento de aplicações, o *SIMCOP* poderá ser utilizado como ferramenta de pesquisa para desenvolvimento e testes de novas métricas de similaridade.

A pesquisa com o ReBaSS, apresentada na seção 5.3.1, foi publicada no Simpósio Brasileiro de Informática na Educação (SBIE/2013), realizado em Campinas/SP, conforme referencia a seguir:

WIEDEMANN, Tiago ; BARBOSA, Jorge L. V. ; RIGO, Sandro J. . Um Modelo para Recomendação de Objetos de Aprendizagem Baseado em Similaridade de Sessões. In: XXIV Simpósio Brasileiro de Informática da Educação (SBIE), 2013, Campinas. Anais do XXIV SBIE. Porto Alegre: SBC, 2013. p. 878-887. DOI: 10.5753/CBIE.SBIE.2013.878

Este artigo obteve a segunda colocação no evento, sendo convidado para uma edição especial da Revista Brasileira de Informática na Educação (RBIE). No SBIE 2013 foram submetidos 360 artigos, sendo que 88 foram aprovados para publicação.

6.3 Trabalhos Futuros

Como resultado das avaliações realizadas com o protótipo implementado foram identificadas duas questões que ainda dependem de aperfeiçoamento. A primeira delas refere-se a questão do alinhamento de metadados ilustrada na Tabela 10 do Capítulo 5. Para tratar esta situação, poderá será necessário a elaboração de algum mecanismo através do qual o *framework* possa identificar grupos de predicados que deverão ser avaliados em conjunto, ao invés de individualmente. Outra questão diz respeito ao tratamento de dados textuais. Para isto poderá ser necessário a implementação de técnicas como o Processamento de Linguagem Natural (LIDDY, 2001) na forma de funções de análise de similaridade de contextos ou atributos.

Outra questão pendente é o uso de ontologias para a análise de similaridade semântica entre contextos ou seus atributos. Para isto, inicialmente deverão ser implementados métodos de leitura e processamento de ontologias descritas em RDF ou OWL, bem como mecanismos que permitam mapear predicados individuais ou em conjunto a conceitos de ontologias. A partir deste mapeamento, o *framework* será capaz identificar cada atributo ou cada contexto como uma instância de determinada classe da ontologia. A partir dai poderão ser aplicadas técnicas como as descritas na seção 2.7.3 para identificar as relações entre contextos ou atributos através das classes as quais estão vinculadas.

No que se refere a avaliação do modelo, ainda poderão ser realizados experimentos visando a identificação de similaridade entre trajetórias, baseado no atributo *Location* dos contextos. Avaliações de performance também poderão ser realizadas, utilizando-se grandes massas de dados, a fim de aperfeiçoar-se os tempos de resposta do *framework*. O protótipo poderá ainda ser aplicado para implementar as aplicações descritas nos trabalhos relacionados (Capítulo 3), visando a comparação de resultados obtidos.

Por fim o SIMCOP prevê suporte, através da classe *SIMCOPValue*, ao recebimento de dados binários. Através deste suporte, poderá ser possível o desenvolvimento de funções capazes de analisar a similaridade em conteúdo multimídia como imagens, vídeos e sons bem como em formatos de arquivos específicos como planilhas e documentos ou ainda dados recebidos de sensores ou via portas USB, por exemplo.

REFERÊNCIAS

ABE, H.; TSUMOTO, S. Detecting Similarity of Transferring Datasets Based on Features of Classification Rules. In: DATA MINING WORKSHOPS, 2009. ICDMW '09. IEEE INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 412–415.

ABRAHAM, S.; LAL, P. S. Spatio-temporal similarity of web user session trajectories and applications in dark web research. In: PACIFIC ASIA CONFERENCE ON INTELLIGENCE AND SECURITY INFORMATICS, 6., 2011, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2011. p. 1–14. (PAISI'11).

ABRAHAM, S.; LAL, P. S. Spatio-temporal similarity of network-constrained moving object trajectories using sequence alignment of travel locations. **Transportation Research Part C: Emerging Technologies**, [S.l.], v. 23, n. 0, p. 109 – 123, 2012. Data Management in Vehicular Networks.

AIUBE, F. A. L. **Econometria para séries financeiras**. [S.l.]: Pontifícia Universidade Católica do Rio de Janeiro, 2007. Working Paper, disponível em http://www.ind.puc-rio.br/UserFiles/File/aiube/Econometria%20Series%20Financeiras.pdf. Acessado em 22 de junho de 2013.

AL-KHALIFA, H. S. Building an Arabic learning object repository with an ad hoc recommendation engine. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 10., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p. 390–394. (iiWAS '08).

ANAND, D.; BHARADWAJ, K. K. Utilizing various sparsity measures for enhancing accuracy of collaborative recommender systems based on local and global similarities. **Expert Systems with Applications**, [S.l.], v. 38, n. 5, p. 5101 – 5109, 2011.

ATEV, S.; MILLER, G.; PAPANIKOLOPOULOS, N. Clustering of Vehicle Trajectories. **Intelligent Transportation Systems, IEEE Transactions on**, [S.l.], v. 11, n. 3, p. 647–657, 2010.

BAKALOV, P.; KEOGH, E.; TSOTRAS, V. J. TS2-tree - an efficient similarity based organization for trajectory data. In: ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 15., 2007, New York, NY, USA. **Proceedings...** ACM, 2007. p. 58:1–58:4. (GIS '07).

BAO, J.; ZHENG, Y.; MOKBEL, M. F. Location-based and preference-aware recommendation using sparse geo-social networking data. In: INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 20., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 199–208. (SIGSPATIAL '12).

BARRENA, M.; JURADO, E.; MáRQUEZ-NEILA, P.; PACHóN, C. A flexible framework to ease nearest neighbor search in multidimensional data spaces. **Data & Knowledge Engineering**, [S.l.], v. 69, n. 1, p. 116 – 136, 2010.

BASHON, Y.; NEAGU, D.; RIDLEY, M. A framework for comparing heterogeneous objects: on the similarity measurements for fuzzy, numerical and categorical attributes. **Soft Computing**, [S.l.], p. 1–21, 2013.

BAYIR, M. A.; DEMIRBAS, M.; EAGLE, N. Mobility profiler: a framework for discovering mobility profiles of cell phone users. **Pervasive and Mobile Computing**, [S.l.], v. 6, n. 4, p. 435 – 454, 2010. Human Behavior in Ubiquitous Environments: Modeling of Human Mobility Patterns.

BECK, A.; RISAGER, C.; ANDERSEN, N.; RAVN, O. Spacio-temporal situation assessment for mobile robots. In: INFORMATION FUSION (FUSION), 2011 PROCEEDINGS OF THE 14TH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 1–8.

BERNDT, D.; CLIFFORD, J. Using dynamic time warping to find patterns in time series. In: AAAI-94 WORKSHOP ON KNOWLEDGE DISCOVERY IN DATABASES, 1994. **Proceedings...** [S.l.: s.n.], 1994. p. 229–248.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The Semantic Web. **Scientific American**, [S.l.], May 2001.

BOGORNY, V.; RENSO, C.; AQUINO, A. R. de; LUCCA SIQUEIRA, F. de; ALVARES, L. O. CONSTAnT – A Conceptual Data Model for Semantic Trajectories of Moving Objects. **Transactions in GIS**, [S.l.], p. n/a–n/a, 2013.

BRAGA, R. B.; COSTA, S. d. M. Medeiros da; MARTIN, H. A trajectory correlation algorithm based on users' daily routines. In: ACM SIGSPATIAL INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 19., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p. 501–504. (GIS '11).

BRAGA, R. B.; TAHIR, A.; BERTOLOTTO, M.; MARTIN, H. A multi-layer data representation of trajectories in social networks based on points of interest. In: WEB INFORMATION AND DATA MANAGEMENT, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 19–26. (WIDM '12).

BROWN, P. J. The Stick-e document: a framework for creating context-aware applications. In: ELECTRONIC PUBLISHING 96, 1996, Laxenburg, Austria. **Proceedings...** IFIP, 1996.

BROWN, P. J.; BOVEY, J. D.; CHEN, X. Context-aware applications: from the laboratory to the marketplace. **IEEE Personal Communications**, [S.l.], v. 4, n. 5, p. 58–64, October 1997.

BUENO, R. d. L. d. S. **Econometria de Séries Temporais**. São Paulo, SP: Cengage Learning, 2008. 299 p.

BUSH, V.; WANG, J. As we may think. **Atlantic Monthly**, [S.l.], v. 176, p. 101–108, 1945.

CALDERARA, S.; PRATI, A.; CUCCHIARA, R. Mixtures of von Mises Distributions for People Trajectory Shape Analysis. **Circuits and Systems for Video Technology, IEEE Transactions on**, [S.l.], v. 21, n. 4, p. 457–471, 2011.

CHA, S.-H. Comprehensive Survey on Distance/Similarity Measures between Probability Density Functions. **International Journal of Mathematical Models and Methods in Applied Sciences**, [S.l.], v. 1, n. 4, p. 300–307, 2007.

CHEN, H.; FININ, T.; JOSHI, A. An Ontology for Context-Aware Pervasive Computing Environments. **Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review**, [S.l.], v. 18, p. 197–207, 2003.

CHEN, L.; NUGENT, C.; WANG, H. A Knowledge-Driven Approach to Activity Recognition in Smart Homes. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. 24, n. 6, p. 961–974, 2012.

CHO, M.; CHOI, C.; KIM, P. Measuring Similarity between Trajectories using Motion Verbs in Semantic Level. In: ADVANCED COMMUNICATION TECHNOLOGY, THE 9TH INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. v. 1, p. 511–515.

COMPUTING SEMANTIC RELATEDNESS USING WIKIPEDIA-BASED EXPLICIT SEMANTIC ANALYSIS, 2007. **Anais...** [S.l.: s.n.], 2007.

COUTAZ, J.; DEARLE, A.; DUPUY-CHESSA, S.; KIRBY, G. N. C.; LACHENAL, C.; MORRISON, R.; REY, G.; ZIRRINTSIS. Working document on gloss ontology. **Tech Rep., Global Smart Spaces Project**, [S.l.], 2003.

CUI, B.; ZHAO, Z.; TOK, W. H. A Framework for Similarity Search of Time Series Cliques with Natural Relations. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. 24, n. 3, p. 385–398, 2012.

D'ACIERNO, A.; LEONE, M.; SAGGESE, A.; VENTO, M. A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries. In: INTELLIGENT TRANSPORTATION SYSTEMS (ITSC), 2012 15TH INTERNATIONAL IEEE CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 989–994.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. **Human-Computer Interaction**, [S.l.], v. 16, n. 2, p. 97–166, December 2001.

DEY, A. K.; ABOWD, G. D.; WOOD, A. CyberDesk: a framework for providing self-integrating context-aware services. In: IUI '98 PROCEEDINGS OF THE 3RD INTERNATIONAL CONFERENCE ON INTELLIGENT USER INTERFACES, 1998, New York, NY. **Anais...** ACM, 1998. p. 47–54.

DEY, D.; SARKAR, S.; DE, P. A distance-based approach to entity reconciliation in heterogeneous databases. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. 14, n. 3, p. 567–582, 2002.

DING, H.; TRAJCEVSKI, G.; SCHEUERMANN, P. Efficient Similarity Join of Large Sets of Moving Object Trajectories. In: TEMPORAL REPRESENTATION AND REASONING, 2008. TIME '08. 15TH INTERNATIONAL SYMPOSIUM ON, 2008. **Anais...** [S.l.: s.n.], 2008. p. 79–87.

DOULAMIS, A.; PELEKIS, N.; THEODORIDIS, Y. EasyTracker: an android application for capturing mobility behavior. In: INFORMATICS (PCI), 2012 16TH PANHELLENIC CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 357–362.

DRIVER, C.; CLARKE, S. Context-aware trails [mobile computing]. **Computer**, [S.l.], v. 37, n. 8, p. 97–99, August 2004.

DRIVER, C.; CLARKE, S. An application framework for mobile, context-aware trails. **Pervasive Mob. Comput.**, Amsterdam, The Netherlands, The Netherlands, v. 4, n. 5, p. 719–736, Oct. 2008.

ENCYCLOPEDIA BRITANNICA. **Ontology**. Acessado em 03 de Abril de 2013, http://www.britannica.com/EBchecked/topic/429409/ontology.

ESLING, P.; AGON, C. Time-series data mining. **ACM Comput. Surv.**, New York, NY, USA, v. 45, n. 1, p. 12:1–12:34, Dec. 2012.

EVANS, M. R.; OLIVER, D.; SHEKHAR, S.; HARVEY, F. Summarizing trajectories into k-primary corridors: a summary of results. In: INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 20., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 454–457. (SIGSPATIAL '12).

FERSCHA, A.; HECHINGER, M.; RIENER, A.; SCHMITZBERGER, H.; FRANZ, M.; ROCHA, M.; ZEIDLER, A. Context-Aware Profiles. In: AUTONOMIC AND AUTONOMOUS SYSTEMS, 2006. ICAS '06. 2006 INTERNATIONAL CONFERENCE ON, 2006. **Anais...** [S.l.: s.n.], 2006. p. 48–48.

FISCHER, Y.; BAUER, A.; BEYERER, J. A conceptual framework for automatic situation assessment. In: COGNITIVE METHODS IN SITUATION AWARENESS AND DECISION SUPPORT (COGSIMA), 2011 IEEE FIRST INTERNATIONAL MULTI-DISCIPLINARY CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 234–239.

FRANKLIN, D.; FLASCHBART, J. All gadget and no representation makes jack a dull environment. In: AAAI 98 SPRING SYMPOSIUM ON INTELLIGENT ENVIRONMENTS, 1998, Menlo Park, CA. **Proceedings...** AAAI Press, 1998.

FU, T. chung. A review on time series data mining. **Engineering Applications of Artificial Intelligence**, [S.l.], v. 24, n. 1, p. 164 – 181, 2011.

GAN, M.; DOU, X.; WANG, D.; JIANG, R. DOPCA: a new method for calculating ontology-based semantic similarity. In: COMPUTER AND INFORMATION SCIENCE (ICIS), 2011 IEEE/ACIS 10TH INTERNATIONAL CONFERENCE ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 110–115.

GARZON, S. R.; HRITSEVSKYY, D. Model-based generation of scenario-specific event sequences for the simulation of recurrent user behavior within context-aware applications (WIP). In: SYMPOSIUM ON THEORY OF MODELING AND SIMULATION - DEVS INTEGRATIVE M&S SYMPOSIUM, 2012., 2012, San Diego, CA, USA. **Proceedings...** Society for Computer Simulation International, 2012. p. 29:1–29:6. (TMS/DEVS '12).

GHAOUI, L. E. **Bag-of-words representation of text**. Acessado em 18 de Julho de 2013, http://inst.eecs.berkeley.edu/~ee127a/book/login/exa_vecs_bag_of_words_rep.html.

GICQUEL, P.; LENNE, D. Using Semantic Similarities to Instrument Informal Learning Activities in Ubiquitous Environments. In: ADVANCED LEARNING TECHNOLOGIES (ICALT), 2012 IEEE 12TH INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 618–620.

GOLDSCHMIDT, R.; PASSOS, E. **Data mining**: um guia prático. 4. ed. Rio de Janeiro, RJ: Elsevier, 2005.

GRECO, G.; TERRACINA, G. Frequency-based similarity for parameterized sequences: formal framework, algorithms, and applications. **Inf. Sci.**, New York, NY, USA, v. 237, p. 176–195, July 2013.

GRUBER, T. Ontology. **Entry in the Encyclopedia of Database Systems, Ling Liu and M. Tamer Özsu (Eds.)**, [S.l.], Sept. 2008.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. **International Journal of Human-Computer Studies**, [S.l.], v. 43, n. 5–6, p. 907 – 928, 1995.

GRéGOIRE, N.; BOUILLOT, M. **Hausdorff distance between convex polygons**. Acessado em 16 de Julho de 2013, http://cgm.cs.mcgill.ca/~godfried/teaching/cg-projects/98/normand/main.html.

GU, T.; WANG, X. H.; PUNG, H. K.; ZHANG, D. Q. An Ontology-based Context Model in Intelligent Environments. In: IN PROCEEDINGS OF COMMUNICATION NETWORKS AND DISTRIBUTED SYSTEMS MODELING AND SIMULATION CONFERENCE, 2004. **Anais...** [S.l.: s.n.], 2004. p. 270–275.

GUDMUNDSSON, J.; THOM, A.; VAHRENHOLD, J. Of motifs and goals: mining trajectory data. In: INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 20., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 129–138. (SIGSPATIAL '12).

GUNOPULOS, D.; TRAJCEVSKI, G. Similarity in (spatial, temporal and) spatio-temporal datasets. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 15., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 554–557. (EDBT '12).

GUY, I.; JACOVI, M.; PERER, A.; RONEN, I.; UZIEL, E. Same places, same things, same people?: mining user similarity on social media. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 2010., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 41–50. (CSCW '10).

HAMID, R.; MADDI, S.; JOHNSON, A.; BOBICK, A.; ESSA, I.; ISBELL, C. A novel sequence representation for unsupervised analysis of human activities. **Artif. Intell.**, Essex, UK, v. 173, n. 14, p. 1221–1244, Sept. 2009.

HAMILTON, J. **The Time Series Analysis**. [S.l.]: Princeton University Press, 1994.

HAN, J.; KAMBER, M. **Data Mining**: concepts and techniques. 2. ed. 500 Sansome Street, Suite 400, San Francisco, CA 94111: Elsevier, 2006.

HAO, Q.; CAI, R.; WANG, C.; XIAO, R.; YANG, J.-M.; PANG, Y.; ZHANG, L. Equip tourists with knowledge mined from travelogues. In: WORLD WIDE WEB, 19., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 401–410. (WWW '10).

HECKMANN, D. A Specialized Representation for Ubiquitous Computing and User Modeling. In: FIRST WORKSHOP ON USER MODELLING FOR UBIQUITOUS COMPUTING AT UM, 2003. **Proceedings...** [S.l.: s.n.], 2003.

HECKMANN, D. **Ubiquitous User Modeling**. 2005. Tese (Doutorado em Ciência da Computação) — Universitat des Saarlandes, 2005.

HIDASI, B.; TIKK, D. Enhancing matrix factorization through initialization for implicit feedback databases. In: WORKSHOP ON CONTEXT-AWARENESS IN RETRIEVAL AND RECOMMENDATION, 2., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 2–9. (CaRR '12).

HOLANDA FERREIRA, A. B. de. **Aurélio**: o dicionário da língua portuguesa. 2. ed. [S.l.]: Ed. Positivo, 2008.

HONG, W.; LI, L.; LI, T. Product recommendation with temporal dynamics. **Expert Syst. Appl.**, Tarrytown, NY, USA, v. 39, n. 16, p. 12398–12406, Nov. 2012.

HORROCKS, I.; PATEL-SCHNEIDER, P. F.; MCGUINNESS, D. L.; WELTY, C. A. OWL: a description logic based ontology language for the semantic web. In: BAADER, F.; CALVANESE, D.; MCGUINNESS, D.; NARDI, D.; PATEL-SCHNEIDER, P. F. (Ed.). **The Description Logic Handbook**: theory, implementation, and applications (2nd edition). [S.l.]: Cambridge University Press, 2007.

HU, X.; XU, P.; WU, S.; ASGARI, S.; BERGSNEIDER, M. A data mining framework for time series estimation. **J. of Biomedical Informatics**, San Diego, USA, v. 43, n. 2, p. 190–199, Apr. 2010.

ILYAS, M.; KüNG, J. A comparative analysis of similarity measurement techniques through SimReq framework. In: INTERNATIONAL CONFERENCE ON FRONTIERS OF INFORMATION TECHNOLOGY, 7., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p. 47:1–47:6. (FIT '09).

ILYAS, M.; KUNG, J. Ontology-Based Similarity Measurement in Software Projects through SimReq Framework. In: SOFTWARE ENGINEERING ADVANCES (ICSEA), 2010 FIFTH INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 412–416.

JAGADEESH CHANDRA BOSE, R. P.; AALST, W. M. P. van der. Process diagnostics using trace alignment: opportunities, issues, and challenges. **Inf. Syst.**, Oxford, UK, UK, v. 37, n. 2, p. 117–141, Apr. 2012.

JIANG, Y.; DENG, D.; WANG, J.; LI, G.; FENG, J. Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints. In: JOINT EDBT/ICDT 2013 WORKSHOPS, 2013, New York, NY, USA. **Proceedings...** ACM, 2013. p. 341–348. (EDBT '13).

JUNIOR, J. A. **Estudo da influência de diversas medidas de similaridade na previsão de séries temporais utilizando o algoritmo KNN-TSP**. 2012. Dissertação (Mestrado em Ciência da Computação) — UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ, FOZ DO IGUAÇU, PR, 2012.

LEE, S.; KANG, S. Clustering navigation sequences to create contexts for guiding code navigation. **Journal of Systems and Software**, [S.l.], n. 0, p. –, 2013.

LEE, W.-C.; SI, W.; CHEN, L.-J.; CHEN, M. C. HTTP: a new framework for bus travel time prediction based on historical trajectories. In: INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 20., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 279–288. (SIGSPATIAL '12).

LI, J.; SONG, Y.; WANG, K. Web User's Access Path Clustering. In: NETWORKING, SENSING AND CONTROL, 2007 IEEE INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. p. 111–114.

LI, Q.; ZHENG, Y.; XIE, X.; CHEN, Y.; LIU, W.; MA, W.-Y. Mining user similarity based on location history. In: ACM SIGSPATIAL INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 16., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p. 34:1–34:10. (GIS '08).

LI, W.; EICKHOFF, C.; VRIES, A. P. de. Want a coffee?: predicting users' trails. In: ACM SIGIR CONFERENCE ON RESEARCH AND DEVELOPMENT IN INFORMATION RETRIEVAL, 35., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 1171–1172. (SIGIR '12).

LI, X.; CEIKUTE, V.; JENSEN, C.; TAN, K. Effective Online Group Discovery in Trajectory Databases. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. PP, n. 99, p. 1–1, 2012.

LIAO, Z.; JIANG, D.; CHEN, E.; PEI, J.; CAO, H.; LI, H. Mining Concept Sequences from Large-Scale Search Logs for Context-Aware Query Suggestion. **ACM Trans. Intell. Syst. Technol.**, New York, NY, USA, v. 3, n. 1, p. 17:1–17:40, Oct. 2011.

LIDDY, E. D. **Encyclopedia of Library and Information Science**: natural language processing. 2. ed. New York: Marcel Decker, Inc., 2001.

LIU, Y.; SHAO, Z. The Similarity Calculation of Concepts from Different Ontologies Based on Cosine. In: INFORMATION MANAGEMENT, INNOVATION MANAGEMENT AND INDUSTRIAL ENGINEERING (ICIII), 2010 INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. v. 4, p. 130–134.

LUCKHAM, D. C. **The Power of Events**: an introduction to complex event processing in distributed enterprise systems. Boston, MA, USA: Addison-Wesley, 2001.

LV, M.; CHEN, L.; CHEN, G. Mining user similarity based on routine activities. **Inf. Sci.**, New York, NY, USA, v. 236, p. 17–32, July 2013.

MA, C.; LU, H.; SHOU, L.; CHEN, G. KSQ: top-k similarity query on uncertain trajectories. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. PP, n. 99, p. 1–1, 2012.

MA, H.; CAO, H.; YANG, Q.; CHEN, E.; TIAN, J. A habit mining approach for discovering similar mobile users. In: WORLD WIDE WEB, 21., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 231–240. (WWW '12).

MASCIARI, E. Finding homogeneous groups in trajectory streams. In: THIRD ACM SIGSPATIAL INTERNATIONAL WORKSHOP ON GEOSTREAMING, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 11–18. (IWGS '12).

MASCIARI, E. Warehousing and querying trajectory data streams with error estimation. In: DATA WAREHOUSING AND OLAP, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 113–120. (DOLAP '12).

MORAES, A. F. de; BASTOS, L. C. Framework of integration for collaboration and spatial data mining among heterogeneous sources in the web. In: ACM SIGSPATIAL INTERNATIONAL WORKSHOP ON DATA MINING FOR GEOINFORMATICS, 1., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 19–28. (DMG '10).

MOTIK, B.; SHEARER, R.; HORROCKS, I. Hypertableau Reasoning for Description Logics. **Journal of Artificial Intelligence Research**, [S.l.], v. 36, p. 165–228, Sept. 2009.

MURAYAMA, H.; YAMADA, K. Detection of unusual human activity based on sequence of actions with MHI and CDP. In: TENCON 2010 - 2010 IEEE REGION 10 CONFERENCE, 2010. **Anais...** [S.l.: s.n.], 2010. p. 1663–1667.

NAUDET, Y. Reconciling Context, Observations and Sensors in Ontologies for Pervasive Computing. In: SEMANTIC MEDIA ADAPTATION AND PERSONALIZATION (SMAP), 2011 SIXTH INTERNATIONAL WORKSHOP ON, 2011. **Anais...** [S.l.: s.n.], 2011. p. 3–8.

OASIS - Organization for the Advancement of Structured Information Standards. **OASIS HumanMarkup TC**. Retrieved 10 April, 2013, `https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=humanmarkup`

OBWEGER, H.; SUNTINGER, M.; SCHIEFER, J.; RAIDL, G. Similarity searching in sequences of complex events. In: RESEARCH CHALLENGES IN INFORMATION SCIENCE (RCIS), 2010 FOURTH INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. p. 631–640.

OH, J.; YU, H. iSampling: framework for developing sampling methods considering user's interest. In: ACM INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT, 21., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 1667–1671. (CIKM '12).

ORWANT, J. Apprising the user of user models: doppelgänger's interface. In: FORTH INTERNATIONAL CONFERENCE ON USER MODELING, 1994, Massaschusetts, USA. **Proceedings...** [S.l.: s.n.], 1994. p. 151–156.

PANAGIOTAKIS, C.; PELEKIS, N.; KOPANAKIS, I.; RAMASSO, E.; THEODORIDIS, Y. Segmentation and Sampling of Moving Object Trajectories Based on Representativeness. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. 24, n. 7, p. 1328–1343, 2012.

PARZEN, E. On Estimation of a Probability Density Function and Mode. **The Annals of Mathematical Statistics**, [S.l.], v. 33, n. 3, p. 1065–1076, sep 1962.

PASCOE, J.; RYAN, N. S.; MORSE, D. R. Human Computer Giraffe Interaction: HCI in the Field. In: WORKSHOP ON HUMAN COMPUTER INTERACTION WITH MOBILE DEVICES, 1998. **Anais...** [S.l.: s.n.], 1998.

PELEKIS, N.; KOPANAKIS, I.; NTOUTSI, I.; MARKETOS, G.; THEODORIDIS, Y. Mining Trajectory Databases via a Suite of Distance Operators. In: DATA ENGINEERING WORKSHOP, 2007 IEEE 23RD INTERNATIONAL CONFERENCE ON, 2007. **Anais...** [S.l.: s.n.], 2007. p. 575–584.

PETITJEAN, F.; KETTERLIN, A.; GANçARSKI, P. A global averaging method for dynamic time warping, with applications to clustering. **Pattern Recogn.**, New York, NY, USA, v. 44, n. 3, p. 678–693, Mar. 2011.

PINHO, D.; VIVACQUA, A.; MEDEIROS, S.; SOUZA, J. de. Ontology Based Design for Similarity Discovery in SYMBAD Project. In: COMPUTER SUPPORTED COOPERATIVE

WORK IN DESIGN, 2006. CSCWD '06. 10TH INTERNATIONAL CONFERENCE ON, 2006. **Anais...** [S.l.: s.n.], 2006. p. 1–6.

PREE, W. Meta Patterns: a means for capturing the essentials of reusable object-oriented design. In: EUROPEAN CONFERENCE ON OBJECT-ORIENTED PROGRAMMING, 8., 1994. **Proceedings...** Springer-Verlag, 1994. p. 150–162.

PRESSMAN, R. **Engenharia de Software**. 6. ed. [S.l.]: Mc Graw Hill, 2006.

QIN, W.; SHI, Y.; SUO, Y. Ontology-based context-aware middleware for smart spaces. **Tsinghua Science and Technology**, [S.l.], v. 12, n. 6, p. 707–713, 2007.

RANGANATHAN, A.; MCGRATH, R. E.; CAMPBELL, R. H.; MICKUNAS, M. D. Use of ontologies in a pervasive computing environment. **Knowl. Eng. Rev.**, [S.l.], v. 18, n. 3, p. 209–220, 2003.

RENAUD, K.; GRAY, P. Making sense of low-level usage data to understand user activities. In: SOUTH AFRICAN INSTITUTE OF COMPUTER SCIENTISTS AND INFORMATION TECHNOLOGISTS ON IT RESEARCH IN DEVELOPING COUNTRIES, 2004., 2004, Republic of South Africa. **Proceedings...** South African Institute for Computer Scientists and Information Technologists, 2004. p. 115–124. (SAICSIT '04).

RODDEN, T.; CHEVERST, K.; DAVIES, K.; DIX, A. Exploiting context in HCI design for mobile systems. In: WORKSHOP ON HUMAN COMPUTER INTERACTION WITH MOBILE DEVICES, 1998, Glasgow, Scotland. **Proceedings...** [S.l.: s.n.], 1998.

ROSA, J. H. da. **ORACON**: an adaptive model for contexts prediction. 2013. Dissertação (Mestrado em Ciência da Computação) — Universidade do Vale do Rio dos Sinos, 2013.

SARANGI, S. R.; MURTHY, K. DUST: a generalized notion of similarity between uncertain time series. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 16., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 383–392. (KDD '10).

SATYANARAYANAN, M. Pervasive Computing: vision and challenges. **IEEE Personal Communications**, [S.l.], v. 8, p. 10–17, 2001.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. In: MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 1994. PROCEEDINGS., WORKSHOP ON, 1994. **Anais...** [S.l.: s.n.], 1994. p. 85–90.

SCHILIT, B. N.; THEIMER, M. M. Disseminating active map information to mobile hosts. **IEEE Network**, [S.l.], v. 8, n. 5, p. 22–32, Sept. 1994.

SELVAKUMAR, A.; SUDHA, R. An approach of concept similarity computation in taxonomy ontology. In: INTELLIGENT AGENT MULTI-AGENT SYSTEMS, 2009. IAMA 2009. INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 1–4.

SHARIATMADARI, S.; MAMAT, A.; IBAHIM, H.; MUSTAPHA, N. Discovering semantic similarity association in semantic search system. In: INTERNATIONAL CONFERENCE ON INFORMATION INTEGRATION AND WEB-BASED APPLICATIONS & SERVICES, 11., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p. 331–337. (iiWAS '09).

SHARMA, A.; PUJARI, A. K.; PALIWAL, K. K. Intrusion detection using text processing techniques with a kernel based similarity measure. **Computers & Security**, [S.l.], v. 26, n. 7–8, p. 488 – 495, 2007.

SHIKIDA, P. F. A.; MARGARIDO, M. A. **Uma análise econométrica de sazonalidade dos preços da cana-de-açúcar, estado do Paraná, 2011-2007**. Acessado em 11 de Junho de 2013, `ftp://ftp.sp.gov.br/ftpiea/publicacoes/IE/2009/tec7-0209.pdf`.

SILVA, J. M.; ROSA, J. H.; BARBOSA, J. L.; BARBOSA, D. N.; PALAZZO, L. A. Content distribution in trail-aware environments. **Journal of the Brazilian Computer Society**, [S.l.], v. 16, n. 3, p. 163–176, 2010.

SIRIN, E.; PARSIA, B.; GRAU, B. C.; KALYANPUR, A.; KATZ, Y. Pellet: a practical owl-dl reasoner. **Web Semantics: Science, Services and Agents on the World Wide Web**, [S.l.], v. 5, n. 2, p. 51 – 53, 2007. Software Engineering and the Semantic Web.

SKOPAL, T.; BUSTOS, B. On nonmetric similarity search problems in complex domains. **ACM Comput. Surv.**, New York, NY, USA, v. 43, n. 4, p. 34:1–34:50, oct 2011.

SLANEY, J.; THIéBAUX, S. Blocks World revisited. **Artificial Intelligence**, [S.l.], v. 125, n. 1–2, p. 119 – 153, 2001.

SMITH, A. **Who Controls the Past Controls the Future - Life Annotation in Principle and Practice**. 2008. Tese (Doutorado em Ciência da Computação) — University of Southampton, 2008.

SOUZA, R. C. **Modelos Estruturais para Previsão de Séries Temporais** : abordagens clássica e bayesiana. [S.l.]: IMPA, 1989.

SPENCE, M.; DRIVER, C.; CLARKE, S. Sharing Context History in Mobile, Context-Aware Trails-Based Applications. In: ECHISE 2005), 1., 2005, Munich, Germany. **Anais...** [S.l.: s.n.], 2005.

STRANG, T.; LINNHOFF-POPIEN, C.; FRANK, K. Applications of a Context Ontology Language. In: UNIVERSITY OF SPLIT, CROATIA, 2003. **Anais...** [S.l.: s.n.], 2003. p. 14–18.

THAKUR, G. S.; HELMY, A.; HSU, W.-J. Similarity analysis and modeling in mobile societies: the missing link. In: ACM WORKSHOP ON CHALLENGED NETWORKS, 5., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 13–20. (CHANTS '10).

THEODORIDIS, Y.; SILVA, J.; NASCIMENTO, M. On the Generation of Spatiotemporal Datasets. In: GüTING, R.; PAPADIAS, D.; LOCHOVSKY, F. (Ed.). **Advances in Spatial Databases**. [S.l.]: Springer Berlin Heidelberg, 1999. p. 147–164. (Lecture Notes in Computer Science, v. 1651).

THIAGARAJAN, R.; MANJUNATH, G.; STUMPTNER, M. Computing semantic similarity using ontologies. In: ISWC 08, THE INTERNATIONAL SEMANTIC WEB CONFERENCE (ISWC), 2008. **Anais...** [S.l.: s.n.], 2008.

THONGKRAU, T.; LALITROJWONG, P. Classifying instances into lexical ontology concepts using latent semantic analysis. In: COMPUTER AND AUTOMATION ENGINEERING (ICCAE), 2010 THE 2ND INTERNATIONAL CONFERENCE ON, 2010. **Anais...** [S.l.: s.n.], 2010. v. 1, p. 66–70.

THORUP, M. Bottom-k and priority sampling, set similarity and subset sums with minimal independence. In: ACM SYMPOSIUM ON SYMPOSIUM ON THEORY OF COMPUTING, 45., 2013, New York, NY, USA. **Proceedings...** ACM, 2013. p. 371–380. (STOC '13).

TSARKOV, D.; HORROCKS, I. FaCT++ Description Logic Reasoner: system description. In: INT. JOINT CONF. ON AUTOMATED REASONING (IJCAR 2006), 2006. **Anais...** Springer, 2006. p. 292–297.

VANATHI, B.; RHYMEND UTHARIARAJ, V. Ontology based service discovery for context aware computing. In: ADVANCED COMPUTING, 2009. ICAC 2009. FIRST INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 51–54.

VISWANATHAN, V.; ILANGO, K. Ranking semantic relationships between two entities using personalization in context specification. **Information Sciences**, [S.l.], v. 207, n. 0, p. 35 – 49, 2012.

VLACHOS, M.; GUNOPOULOS, D.; KOLLIOS, G. Discovering Similar Multidimensional Trajectories. In: INTERNATIONAL CONFERENCE ON DATA ENGINEERING, 18., 2002, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2002. p. 673–. (ICDE '02).

W3C - World Wide Web Consortium. **The W3C Semantic Web Ontologies website**. Retrieved 10 April, 2013, `http://www.w3.org/standards/semanticweb/`.

W3C - World Wide Web Consortium. **OWL Specification**. Retrieved 19 April, 2013, `http://www.w3.org/TR/owl-guide/`.

W3C - World Wide Web Consortium. **The W3C Inference on Semantic Web**. Retrieved 19 April, 2013, `http://www.w3.org/standards/semanticweb/inference`.

WANG, H.; LIN, Z.; MCCLEAN, S.; LIU, J. Measuring Similarity for Multidimensional Sequences. In: IEEE INTERNATIONAL CONFERENCE ON DATA MINING WORKSHOPS, 2010., 2010, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2010. p. 281–287. (ICDMW '10).

WANG, H.; LIU, K. User oriented trajectory similarity search. In: ACM SIGKDD INTERNATIONAL WORKSHOP ON URBAN COMPUTING, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 103–110. (UrbComp '12).

WANG, J.; LI, G.; FENG, J. Can we beat the prefix filtering?: an adaptive framework for similarity join and search. In: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA, 2012., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 85–96. (SIGMOD '12).

WANG, X.; ZHANG, D. Q.; GU, T.; PUNG, H. Ontology based context modeling and reasoning using OWL. In: PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS, 2004. PROCEEDINGS OF THE SECOND IEEE ANNUAL CONFERENCE ON, 2004. **Anais...** [S.l.: s.n.], 2004. p. 18–22.

WARD, A.; JONES, A.; HOPPER, A. A new location technique for the active office. **IEEE Personal Communications**, [S.l.], p. 42–47, October 1997.

WEI, L.-Y.; ZHENG, Y.; PENG, W.-C. Constructing popular routes from uncertain trajectories. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING, 18., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 195–203. (KDD '12).

WEISER, M. The Computer for the 21st Century. **Scientific America**, [S.l.], v. 265, n. 3, p. 94–104, sep 1991.

WIEDEMANN, T.; BARBOSA, J. L. V.; RIGO, S. J. Um Modelo para Recomendação de Objetos de Aprendizagem Baseado em Similaridade de Sessões. In: XXIV SIMPóSIO BRASILEIRO DE INFORMáTICA DA EDUCAçãO (SBIE), 2013, Campinas. **Anais...** Sociedade Brasileira de Computação, 2013. p. 878–887.

WONGSUPHASAWAT, K.; PLAISANT, C.; TAIEB-MAIMON, M.; SHNEIDERMAN, B. Querying event sequences by exact match or similarity search: design and empirical evaluation. **Interact. Comput.**, New York, NY, USA, v. 24, n. 2, p. 55–68, Mar. 2012.

XIAO, X.; ZHENG, Y.; LUO, Q.; XIE, X. Finding similar users using category-based location history. In: SIGSPATIAL INTERNATIONAL CONFERENCE ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 18., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 442–445. (GIS '10).

XUECHENG, L. Entropy, distance measure and similarity measure of fuzzy sets and their relations. **Fuzzy Sets and Systems**, [S.l.], v. 52, n. 3, p. 305 – 318, 1992.

YAN, Z.; CHAKRABORTY, D.; PARENT, C.; SPACCAPIETRA, S.; ABERER, K. SeMiTri: a framework for semantic annotation of heterogeneous trajectories. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 14., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p. 259–270. (EDBT/ICDT '11).

YING, J. J.-C.; LU, E. H.-C.; LEE, W.-C.; WENG, T.-C.; TSENG, V. S. Mining user similarity from semantic trajectories. In: ACM SIGSPATIAL INTERNATIONAL WORKSHOP ON LOCATION BASED SOCIAL NETWORKS, 2., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p. 19–26. (LBSN '10).

YING, J. J.-C.; LU, E. H.-C.; TSENG, V. S. Followee recommendation in asymmetrical location-based social networks. In: ACM CONFERENCE ON UBIQUITOUS COMPUTING, 2012., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 988–995. (UbiComp '12).

YOON, H.; ZHENG, Y.; XIE, X.; WOO, W. Social itinerary recommendation from user-generated digital trails. **Personal Ubiquitous Comput.**, London, UK, UK, v. 16, n. 5, p. 469–484, June 2012.

ZEINALIPOUR-YAZTI, D.; LAOUDIAS, C.; COSTA, C.; VLACHOS, M.; ANDREOU, M. I.; GUNOPULOS, D. Crowdsourced Trace Similarity with Smartphones. **Knowledge and Data Engineering, IEEE Transactions on**, [S.l.], v. 25, n. 6, p. 1240–1253, 2013.

ZHANG, H.; HU, R.; WANG, Y.; LENG, Q.; CHEN, Q. A novel method of similarity search for moving object trajectories. In: AUTOMATIC CONTROL AND ARTIFICIAL INTELLIGENCE (ACAI 2012), INTERNATIONAL CONFERENCE ON, 2012. **Anais...** [S.l.: s.n.], 2012. p. 235–238.

ZHANG, J.; YOU, S.; GRUENWALD, L. U^{2}STRA: high-performance data management of ubiquitous urban sensing trajectories on gpgpus. In: ACM WORKSHOP ON CITY DATA MANAGEMENT WORKSHOP, 2012., 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 5–12. (CDMW '12).

ZHANG, L.; WANG, X.; ZHANG, L.; CHEN, Y.; SHI, Y. Context-based knowledge discovery and its application. In: DATA MINING AND INTELLIGENT KNOWLEDGE MANAGEMENT WORKSHOP, 2012, New York, NY, USA. **Proceedings...** ACM, 2012. p. 7:1–7:9. (DM-IKM '12).

ZHAO, Y.; WANG, N.; DENG, S. The Method of Concept Similarity Computation in Taxonomy Ontology. In: ELECTRONIC COMPUTER TECHNOLOGY, 2009 INTERNATIONAL CONFERENCE ON, 2009. **Anais...** [S.l.: s.n.], 2009. p. 325–328.

ZHENG, V. W.; ZHENG, Y.; XIE, X.; YANG, Q. Towards mobile intelligence: learning from gps history data for collaborative recommendation. **Artif. Intell.**, Essex, UK, v. 184-185, p. 17–37, June 2012.

APÊNDICE A LISTA DOS TRABALHOS ANALISADOS

Para seleção dos trabalhos relacionados descritos no Capítulo 3 foram analisados artigos obtidos através de consultas às bases de periódicos da *IEEE*, *Elsevier*, *ACM* e *Springer-Verlag*. Os artigos listados a seguir não foram considerados como trabalhos relacionados, por não atenderem aos critérios apresentados no Capítulo 3. Alguns trabalhos foram substituídos por outros que tratavam do mesmo tema mas em publicações mais recentes e/ou relevantes.

Apesar de não terem sidos selecionados para comparação, estes trabalhos possuem relação com o tema e contribuiram para formar uma visão mais ampla sobre o problema da análise de similaridade em sequências de contextos. Eles encontram-se listados por data de publicação e agrupados por área temática:

Análise de Hábitos de Usuários

1. *A habit mining approach for discovering similar mobile users* (MA et al., 2012a);
2. *iSampling: framework for developing sampling methods considering user's interest* (OH; YU, 2012);
3. *Model-based generation of scenario-specific event sequences for the simulation of recurrent user behavior within context-aware applications (WIP)* (GARZON; HRITSEVSKYY, 2012);
4. *Equip tourists with knowledge mined from travelogues* (HAO et al., 2010);
5. *Similarity analysis and modeling in mobile societies: the missing link* (THAKUR; HELMY; HSU, 2010);
6. *A novel sequence representation for unsupervised analysis of human activities* (HAMID et al., 2009);
7. *Making sense of low-level usage data to understand user activities* (RENAUD; GRAY, 2004);

Similaridade de situações

1. *Using Semantic Similarities to Instrument Informal Learning Activities in Ubiquitous Environments*(GICQUEL; LENNE, 2012);
2. *Context-based knowledge discovery and its application* (ZHANG et al., 2012b);
3. *A conceptual framework for automatic situation assessment* (FISCHER; BAUER; BEYERER, 2011);
4. *Detection of unusual human activity based on sequence of actions with MHI and CDP* (MURAYAMA; YAMADA, 2010);
5. *Ontology-Based Similarity Measurement in Software Projects through SimReq Framework* (ILYAS; KUNG, 2010);
6. *Detecting Similarity of Transferring Datasets Based on Features of Classification Rules* (ABE; TSUMOTO, 2009);
7. *Context-Aware Profiles* (FERSCHA et al., 2006);

8. *Ontology Based Design for Similarity Discovery in SYMBAD Project* (PINHO et al., 2006);

Similaridade de Sequências

1. *Frequency-based similarity for parameterized sequences: Formal framework, algorithms, and applications* (GRECO; TERRACINA, 2013);
2. *Clustering navigation sequences to create contexts for guiding code navigation*(LEE; KANG, 2013);
3. *Can we beat the prefix filtering?: an adaptive framework for similarity join and search* (WANG; LI; FENG, 2012);
4. *Process diagnostics using trace alignment: Opportunities, issues, and challenges* (JAGADEESH CHANDRA BOSE; AALST, 2012);
5. *Querying event sequences by exact match or similarity search: Design and empirical evaluation* (WONGSUPHASAWAT et al., 2012);
6. *Mining Concept Sequences from Large-Scale Search Logs for Context-Aware Query Suggestion* (LIAO et al., 2011);
7. *Measuring Similarity for Multidimensional Sequences* (WANG et al., 2010);

Similaridade de Trajetórias

1. *CONSTAnT – A Conceptual Data Model for Semantic Trajectories of Moving Objects* (BOGORNY et al., 2013);
2. *KSQ: Top-k Similarity Query on Uncertain Trajectories* (MA et al., 2012b).
3. *A system for storing and retrieving huge amount of trajectory data, allowing spatio-temporal dynamic queries* (D'ACIERNO et al., 2012).
4. *EasyTracker: An Android Application for Capturing Mobility Behavior* (DOULAMIS; PELEKIS; THEODORIDIS, 2012).
5. *A novel method of similarity search for moving object trajectories* (ZHANG et al., 2012a).
6. *Segmentation and Sampling of Moving Object Trajectories Based on Representativeness* (PANAGIOTAKIS et al., 2012).
7. *Similarity in (spatial, temporal and) spatio-temporal datasets* (GUNOPULOS; TRAJCEVSKI, 2012);
8. *HTTP: a new framework for bus travel time prediction based on historical trajectories* (LEE et al., 2012);
9. *U^2STRA: high-performance data management of ubiquitous urban sensing trajectories on GPGPUs* (ZHANG; YOU; GRUENWALD, 2012);
10. *A multi-layer data representation of trajectories in social networks based on points of interest* (BRAGA et al., 2012);
11. *Finding homogeneous groups in trajectory streams* (MASCIARI, 2012a);
12. *Warehousing and querying trajectory data streams with error estimation* (MASCIARI, 2012b);

13. *Constructing popular routes from uncertain trajectories* (WEI; ZHENG; PENG, 2012);
14. *Of motifs and goals: mining trajectory data* (GUDMUNDSSON; THOM; VAHRENHOLD, 2012);
15. *Summarizing trajectories into k-primary corridors: a summary of results* (EVANS et al., 2012);
16. *Location-based and preference-aware recommendation using sparse geo-social networking data*(BAO; ZHENG; MOKBEL, 2012);
17. *Social itinerary recommendation from user-generated digital trails* (YOON et al., 2012);
18. *Towards mobile intelligence: Learning from GPS history data for collaborative recommendation*(ZHENG et al., 2012);
19. *User oriented trajectory similarity search* (WANG; LIU, 2012);
20. *A trajectory correlation algorithm based on users' daily routines* (BRAGA; COSTA; MARTIN, 2011)
21. *SeMiTri: a framework for semantic annotation of heterogeneous trajectories* (YAN et al., 2011);
22. *Spacio-temporal situation assessment for mobile robots* (BECK et al., 2011);
23. *Mixtures of von Mises Distributions for People Trajectory Shape Analysis* (CALDERARA; PRATI; CUCCHIARA, 2011);
24. *Mobility profiler: A framework for discovering mobility profiles of cell phone users*(BAYIR; DEMIRBAS; EAGLE, 2010);
25. *Clustering of Vehicle Trajectories* (ATEV; MILLER; PAPANIKOLOPOULOS, 2010).
26. *Finding similar users using category-based location history* (XIAO et al., 2010);
27. *Framework of integration for collaboration and spatial data mining among heterogeneous sources in the web*; (MORAES; BASTOS, 2010)
28. *Mining user similarity from semantic trajectories*(YING et al., 2010);
29. *Mining user similarity based on location history* (LI et al., 2008);
30. *Efficient Similarity Join of Large Sets of Moving Object Trajectories*(DING; TRAJCEVSKI; SCHEUERMANN, 2008).
31. *TS2-tree - an efficient similarity based organization for trajectory data* (BAKALOV; KEOGH; TSOTRAS, 2007);
32. *Measuring Similarity between Trajectories using Motion Verbs in Semantic Level* (CHO; CHOI; KIM, 2007);
33. *Mining Trajectory Databases via a Suite of Distance Operators* (PELEKIS et al., 2007);
34. *Discovering Similar Multidimensional Trajectories* (VLACHOS; GUNOPOULOS; KOLLIOS, 2002);

Séries Temporais

1. *Enhancing matrix factorization through initialization for implicit feedback databases* (HIDASI; TIKK, 2012);
2. *A data mining framework for time series estimation* (HU et al., 2010);

3. *DUST: a generalized notion of similarity between uncertain time series* (SARANGI; MURTHY, 2010);
4. *Same places, same things, same people?: mining user similarity on social media* (GUY et al., 2010);
5. *A global averaging method for dynamic time warping, with applications to clustering* (PETITJEAN; KETTERLIN; GANçARSKI, 2011).

Outros trabalhos

1. *A framework for comparing heterogeneous objects: on the similarity measurements for fuzzy, numerical and categorical attributes* (BASHON; NEAGU; RIDLEY, 2013);
2. *Efficient parallel partition-based algorithms for similarity search and join with edit distance constraints* (JIANG et al., 2013);
3. *Followee recommendation in asymmetrical location-based social networks* (YING; LU; TSENG, 2012);
4. *Product recommendation with temporal dynamics* (HONG; LI; LI, 2012);
5. *Ranking semantic relationships between two entities using personalization in context specification*; (VISWANATHAN; ILANGO, 2012)
6. *A flexible framework to ease nearest neighbor search in multidimensional data spaces* (BARRENA et al., 2010);
7. *A comparative analysis of Similarity Measurement Techniques through SimReq Framework* (ILYAS; KüNG, 2009);
8. *Intrusion detection using text processing techniques with a kernel based similarity measure* (SHARMA; PUJARI; PALIWAL, 2007);
9. *Web User's Access Path Clustering* (LI; SONG; WANG, 2007);