6
Bibliografia

1. BECHHOFER, S. et al. OWL web ontology language reference. W3C Recommendation, Feb 2004. Disponível em: http://www.w3.org/TR/owl-ref, Último acesso em Mar 2010.
2. BERNSTEIN, P.; MELNIK, S. Model management 2.0: manipulating richer mappings. In Proc. of the 2007 ACM SIGMOD Int'l. Conf. on Management of Data (2007), pp. 1–12.
3. BILKE, A.; NAUMANN, F. Schema matching using duplicates. In Proc. of the 21st Int'l. Conf. on Data Engineering (Apr 2005), pp. 69–80.
4. BRAUNER, D. F.; CASANOVA, M. A.; MILIDIÚ, R. L. Towards gazetteer integration through an instance-based thesauri mapping approach. In Advances in Geoinformatics; VIII Brazilian Symp. on GeoInformatics, GEOINFO (Nov 2007), pp. 235–245.
5. BRAUNER, D. F.; GAZOLA, A.; CASANOVA, M. A. Adaptative matching of database web services export schemas. In Proc. of the 10th Int'l. Conf. on Enterprise Information Systems (ICEIS) (2008), pp. 49–56.
6. BRAUNER, D. F. et al. An instance-based approach for matching export schemas of geographical database Web services. In Proc. of the IX Brazilian Symp. on GeoInformatics (GEOINFO) (2007), pp. 109–120.
7. CASANOVA, M. et al. Database conceptual schema matching. Computer, 40(10):102–104.
8. DO, H.; RAHM E. COMA: a system for flexible combination of schema matching approaches. In Proc. of the 28th Int'l. Conf. on Very Large Data Bases (2002), pp. 610–621.
9. DOAN, A.; DOMINGOS, P.; HALEVY, A. Y. Reconciling schemas of disparate data sources: a machine-learning approach. In Proc. of the 2001 ACM SIGMOD Int'l. Conf. on Management of Data (May 2001), vol. 30, pp. 509–520.
10. EUZENAT, J.; SHVAIKO, P. Ontology matching. Springer-Verlag, 2007.

11. FRAKES, W.; BAEZA-YATES, R. Information retrieval: data structure and algorithms. Prentice Hall, 1992.
12. GAMMA, E. et al. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley Longman Publishing Co., Inc.
13. GOMES, R. V. A.; LEME, L. A. P. P.; CASANOVA, M. A. MatchMaking – a tool to match OWL Schemas.
14. HINDLE, D. Noun classification from predicate-argument structures. In Proc. of the 28th Annual Meeting on Association for Computational Linguistics (1990), pp. 268–275.
15. Klyne, G.; Carroll, J. J.; McBride, B. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, Feb 2004. Disponível em: http://www.w3.org/TR/rdf-concepts, último acesso em Mar 2010.
16. LEE, J. Information retrieval based on conceptual distance in is-a hierarchies. Journal of Documentation *49*, 2 (1993), 188–207.
17. LEME, L. A. P. P. et al. Matching object catalogues. Journal Innovations in Systems and Software Engineering 4, 4 (Oct 2008), 315–328.
18. LEME, L. A. P. P. et al. Database mediation using multi-agent systems. In Proceedings of the 32nd Annual IEEE Software Engineering Workshop, co-located with the 3rd International Symposium on Leveraging Applications of Formal Methods, Verification and Validation (Jan 2009).
19. LEME, L. A. P. P. et al. Instance-based OWL schema matching. In Proceedings of the 11th International Conference on Enterprise Information Systems (May 2009).
20. LEME, L. A. P. P. Conceptual schema matching based on similarity heuristics. Tese (Doutorado), Departamento de Informática, PUC-Rio.
21. LEME, L. A. P. P. et al. Evaluation of similarity measures and heuristics for simple RDF schema matching. Technical Report 44/08, Dept. Informatics, PUC-Rio.
22. LEME, L. A. P. P. et al. Matching object catalogues. J. Innovations in Systems and Software Engineering 4(4), Springer, p. 315–328.
23. LEME, L. A. P. P. et al. Instance-based OWL Schema Matching. Proc. 11th Int'l. Conf. on Enterprise Inf. Systems, Milan, Italy.

24. LIN, D. An information-theoretic definition of similarity. In Proc. of the 15th Int'l. Conf. on Machine Learning (1998), pp. 296–304.
25. MADHAVAN, J.; BERNSTEIN, P. A.; RAHM, E. Generic schema matching with Cupid. In Proc. of the 27th Int'l. Conf. on Very Large Data Bases (2001), pp. 49–58.
26. MADHAVAN, J. et al. Corpus-based schema matching. In Proc. of the 21st Int'l. Conf. on Data Engineering (Apr 2005), pp. 57–68.
27. MADHAVAN, J. et al. Web-scale data integration: You can afford to Pay as You Go. In Proc. of the 3rd Biennial Conf. on Innovative Data Systems Research (CIDR) (2007), vol. 7, pp. 342–350.
28. MELNIK, S.; GARCIA-MOLINA, H.; RAHM, E. Similarity flooding: a versatile graph matching algorithm and its application to schema matching. In Proc. of the 18th Int'l. Conf. on Data Engineering (2002), pp. 117–128.
29. QUINE, W.V. Ontological Relativity. J. of Philosophy, Vol. 65, No. 7, p. 185-212.
30. RAHM, E.; BERNSTEIN, P. A survey of approaches to automatic schema matching. The VLDB Journal 10, 4 (2001), 334–350.
31. RESNIK, P. Using information content to evaluate semantic similarity in a taxonomy. In Proc. of the 14th Int'l. Joint Conf. on Artificial Intelligence (AAAI) (1995), pp. 448–453.
32. TVERSKY, A.; GATI, I. Studies of similarity. Cognition and categorization 1 (1978), 79–98.
33. UDREA, O.; GETOOR, L.; MILLER, R. Leveraging data and structure in ontology integration. In Proceedings of the 2007 ACM SIGMOD Intl'l Conf. on Management of data (Jun 2007), pp. 449–460.
34. WANG, J. et al. Instance-based schema matching for web databases by domain-specific query probing. In Proc. of the 13th Int'l. Conf. on Very Large Data Bases (Aug 2004), pp. 408–419.

Apêndice A.
Lista de comandos

O *Matchmaking* disponibiliza uma série de funcionalidades para que um desenvolvedor possa integrá-lo à sua ferramenta. Abaixo segue a lista com todos os comandos possíveis.

Tabela 5 - Lista de comandos possíveis

Pacote	Comando	Descrição
Schema	GetSchema	Obtém todos os detalhes de um esquema (inclusive todos os seus *Dataset*s) previamente cadastrado no *Matchmaking*.
	GetSchemas	Obtém uma lista com todos os esquemas cadastrados no *Matchmaking*.
	NewSchema	Cadastra um novo esquema no *Matchmaking*. É necessário enviar o arquivo OWL para que o mesmo seja interpretado e validado. Um *dataset* é automaticamente criado representando os dados contidos nesse arquivo.
	NewDataSet	Cadastra um novo *dataset* para um esquema já cadastrado na ferramenta. É importante notar que o *Matchmaking* valida o esquema elemento por elemento para confirmar que se trata do mesmo. Se houver alguma discrepância o sistema informará o erro e não permitirá o cadastro.
Matcher	GetMatcher	Obtém todas as informações de um algoritmo de alinhamento (outros algoritmos utilizados, parâmetros necessários, funções de similaridade utilizadas).
	GetMatchers	Obtém todos os algoritmos existentes na ferramenta. Há uma opção – *showsteps* – que permite obter apenas os algoritmos que não são passos ou todos.
	GetsetTypes	Retorna todos os geradores de conjuntos de dados existentes no sistema.
	NewSetType	Cadastra um novo gerador de conjunto de dados.
	GetSimilarityFunctions	Obtém todas as funções de similaridade registradas no sistema.
	NewSimilarityFunction	Cadastra uma nova função de similaridade.
	NewMatcher	Cadastra um novo algoritmo de alinhamento. É necessário informar um nome, se ele é apenas um passo e todos os passos, funções de similaridade e parâmetros que fará uso.
Execution	GetExecution	Obtém todos os dados de uma determinada execução de um algoritmo.

	GetExecutions	Obtém todas as execuções já realizadas pelo sistema.
	ExecuteMatcher	Executa um algoritmo de alinhamento. Essa é a principal operação do sistema. O solicitante deve apresentar como dados o esquema e *dataset* de origem e destino, que algoritmo planeja utilizar e preencher todos os parâmetros do algoritmo escolhido e de seus passos. Antes de ocorrer a execução, uma validação é feita e, se algum parâmetro não possuir um valor atribuído ou o valor não for do tipo de dados registrado para àquele parâmetro, o sistema informará erro.
	ExecuteBatch	Esse comando permite a execução – através de um arquivo *XML* – de várias operações de alinhamento. O solicitante deve escolher um esquema e *dataset* de origem e de destino e pode informar várias execuções de algoritmos de uma única vez. O solicitante, pode avaliar mais de um algoritmo ou também pode apenas variar os valores dos parâmetros de um algoritmo. Esse comando faz uso do *ExecuteMatcher*.

Apêndice B.
Configuração do Matchmaking

O *Matchmaking* foi criado de forma a ser uma ferramenta altamente personalizável. Apesar da maioria das personalizações poderem ser criadas através da interface gráfica disponível (como a inclusão de novos algoritmos, representações de dados e funções de similaridade), algumas precisam ser informadas antes da inicialização do sistema.

Essas configurações foram embutidas no arquivo *matchmaking.config* (Figura 31), que deve ser instalado no *classpath*[9] da aplicação. Esse arquivo possui algumas configurações que são obrigatórias e outras que variam de acordo com o que existe instalado na ferramenta.

Tomando como exemplo a Figura 31, segue-se uma descrição dos setores apresentados:

- ***Database sector***: é o setor que informa qual a *fábrica* que irá realizar as operações de acesso aos dados. O desenvolvedor deve informar a classe que estende *DAOFactory*.
- ***LOG4J Properties sector***: *LOG4J*[10] é uma infra-estrutura destinada a *logging* de sistemas. Através dessas propriedades (que são as mesmas do arquivo *log4j.properties*) é possível definir qual o grau de registro desejado.
- ***SQL Server DAO Factory sector***: esse é um setor opcional e só deve ser criado se *matchmaking.dao.sqlserver.SQLServerDaoFactory* for utilizada para acesso aos dados. Essa fábrica utiliza um *pool* de conexões para acelerar o processo de obtenção de conexões.
- ***Other data sector***: é o setor que possuem outros dados. Por exemplo, os dados apresentados são utilizados pela representação de dados t*okensOfPropertiesSet*.

[9] *Classpath* é o caminho que a máquina virtual Java usa como informação para o início dos diretórios de classes.
[10] http://logging.apache.org/log4j/

Qualquer função de similaridade, representação de dados ou algoritmo pode fazer uso do arquivo de configuração. Bastando, para isso, escrever a configuração desejada e acessá-la através do método estático *getProperty* da classe *matchmaking.configuration.Configurator*.

```
# Matchmaking's configuration

#--------- Other data sector ---------#

lemmatiserPath = C:/libs/dragon-1.3.3/nlpdata/lemmatiser
stopWordsPath = C:/data/stop-words.txt

#--------- Database sector -----------#

daoFactoryClassName = matchmaking.dao.sqlserver.SQLServerDAOFactory

#--------- SQL Server DAO Factory sector -----------#

jdbcurl = jdbc:sqlserver://localhost\\matchmaking;database=matchmaking;
username = mm
password = $mm*
database = matchmaking
minPoolSize = 5
maxPoolSize = 20
acquireIncrement = 5

#--------- LOG4J Properties sector ---------------#
log4j.rootCategory=WARN, stdout, fileOut

log4j.category.matchmaking=INFO

log4j.appender.stdout=org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%5p [%t] (%F:%L) - %m%n

log4j.appender.fileOut =org.apache.log4j.RollingFileAppender
log4j.appender.fileOut.File=c:/matchmaking.log

log4j.appender.fileOut.MaxFileSize=10000KB

log4j.appender.fileOut.MaxBackupIndex=10

log4j.appender.fileOut.layout=org.apache.log4j.PatternLayout
log4j.appender.fileOut.layout.ConversionPattern=%p %t %c - %m%n
```

Figura 31 - Arquivo de configuração *matchmaking.properties*.