# DESKMatcher

Michael Monych[2][0000−0002−3333−6307], Jan Portisch[1,2][0000−0001−5420−0663], Michael Hladik[2][0000−0002−2204−3138], and Heiko Paulheim[1][0000−0003−4386−8195]

[1] Data and Web Science Group, University of Mannheim, Germany
{jan, heiko}@informatik.uni-mannheim.de
[2] SAP SE Product Engineering Financial Services, Walldorf, Germany
{michael.monych, jan.portisch, michael.hladik}@sap.com

**Abstract.** This paper describes *DESKMatcher*, a label-based ontology matcher. It utilizes background knowledge from the financial services and enterprise domain to better find matches in these domains. The background knowledge utilized for the enterprise domain was in the form of documentation of terms used in SAP software (textual). Therefore, *Word2Vec* and *GloVe* were used for these corpora. The *Financial Industries Business Ontology* (*FIBO*) was used as more specific background knowledge for the financial services domain. Vector space embeddings for this corpus were trained using *RDF2Vec* and *KGloVe*. Individual matchers utilizing one set of embeddings (generated from a combination of method and corpus) are pipelined together after a string-based matchers, searching only for matches between entities that have not been assigned to a match in a previous step. Results on the *OAEI* tracks are expected to be sub-par, because low overlap between corpus and task vocabulary is expected.[3]

**Keywords:** Ontology Matching · Ontology Alignment · Domain Specific Background Knowledge

## 1 Presentation of the System

### 1.1 State, Purpose, General Statement

*DESKMatcher* (Enterprise *D*omain *S*pecific *K*nowledge Matcher) is an element-level, label-based matcher which utilizes vector space embeddings trained by applying multiple techniques on three background knowledge datasets specific to the enterprise and financial services domain, namely the *Financial Industry Business Ontology* (FIBO), the *SAP Glossary*, as well as *SAP Term*. The matcher was implemented for domain-specific matching in the financial services domain where classic schema matching problems are common and can be modelled as ontology matching problems [11].

However, in this paper we evaluate in how far the matcher generalizes to non-business/other domains. The matcher has not been adapted for other tasks.

### 1.2 Specific Techniques Used

The *DESKMatcher* system is implemented as a matching pipeline of subsequent matching steps using multiple domain-specific datasets that were embedded with *RDF2Vec* or *word2vec* depending on their inherent structure. In the following a quick introduction to the datasets used as well as to RDF2Vec will be given.

*External Domain-Specific Datasets* Below, we quickly introduce the sources of background knowledge that have been used:

1. The *Financial Services Business Ontology* (*FIBO*) [3] is used as the most specific source of background knowledge. It is an ontology specific to the financial services domain maintained by the EDM council, with the possibility for outside authors to contribute[4]. The *FIBO* version used contained roughly 88,000 triples with roughly 12,000 unique URIs.
2. The SAP Glossary is a textual corpus describing terms that are relevant for SAP's Enterprise Resource Planning (ERP) software. The resource is not available as ontology but instead in the form of a losely structured text corpus. The glossary was last released in 2017. The set contained definitions for roughly 48,000 terms using roughly 14,000 unique words.
3. The *SAP Term* is larger than the *SAP Glossary* but follows the same objective. It is frequently updated. The resource is not available as ontology but instead in the form of a losely structured text corpus. For this work we used the version as of March 2020. The set contained definitions for roughly 62,000 terms using roughly 16,000 unique words.

*Embedding Approaches Used* In *word2vec* [8] Mikolov et al. present two vector space embedding approaches for textual corpora: *Skip-Gram* (SG) and *continuous bag of words* (CBOW). Embeddings are generated by building a neural network that models randomly drawn context windows given a word (SG) and vice versa (CBOW). *RDF2Vec* [15] is an embedding approach for knowledge graphs that has already been used before in the area of ontology matching [13]. Random walks are generated starting at each node in the knowledge graph. The set of generated walks is then regarded as sentences and a *word2vec* algorithm is applied. Thereby, a vector is obtained for each node and for each edge (that appear in the random walks) in the knowledge graph.[5] *GloVe* [9] is another embedding approach for textual corpora presented 2014 by Pennington et al. Embeddings are generated based on co-occurence probabilities of words in the input corpus. *KGloVe* [2] is an approach to generate embeddings on knowledge graphs presented by Cochez et al. in 2017. Node "co-occurence probabilities" are approximated in a first step, by applying a version of the Bookmark Coloring Algorithm (BCA) [1]. The probabilities are then fed to the standard *GloVe* model, which yields embeddings for each node in the graph. Embeddings for

[4] see `https://github.com/edmcouncil/fibo/blob/master/CONTRIBUTING.md`

[5] More information about RDF2Vec and its application can be found online: `http://rdf2vec.org/`

*FIBO* were trained using the *jrdf2Vec*[6] [12] framework, as well as Cochez et al.'s implemenation of their own *KGloVe* [2]. *SAP Glossary* and *SAP Term* were embedded with *word2vec* (using the *gensim*[7] library [14]) and *GloVe* as made available by Penningotn et al[8].

*Configuration of Embedding approaches* Skip-gram was chosen over *CBOW*. This was based on Mikolov et al.'s results, that *Skip-gram* is better in semantic tasks [8, p. 7], which has also been indicated in [16, p. 4]. Generally, higher dimensions lead to higher performance, however the gain in performance per added dimension seems to greatly decrease after 200 dimensions, wherever dimensions are reported. Therefore the dimensions were fixed at 200[9]. Based on recommended parameter settings from previous work, the *window-size* was fixed to 5, negative sampling with 15 noise words and a smoothing exponent of 0.75 (as per Mikolov et al.'s recommendation in the original paper) was used. The *Skip-gram* embeddings were generated using the implementation in the *gensim* library [14].
The walks required for the *RDF2Vec* model were generated using *jRDF2Vec* [12], while the training of the actual embeddings was conducted using *gensim*'s *Skip-gram* implementation (same as for the text corpora). The walk strategy used to generate walks, is exactly one of the strategies proposed by Ristoski et al. in their original paper (Breadth-first [15]). 100 walks were generated per entity, using a depth of 4, which lead to "sentences" with a maximum length of 12.
To generate the *GloVe* embeddings, the original authors' `C` implementation was used [10]. For *GloVe* three parameters needed to be set: *minCount* was set to 4 in accordance to the value used in *Skip-gram*. *windowsize* was set to 15. $x_{max}$ was set to 10 for this small corpus setting, due to the authors chosing 100 on their large corpus [9].
The implementation by Cochez et al. [11], was used to generate the shuffled co-occurrence files needed as input for the final step of *GloVe*.
Based on their results for best performance, the *PageRank* weighting scheme for context generation would have been chosen, which unfortunately did not execute without fatal errors, even after several attempts to tinker with the code. Therefore the uniform weighting was chosen, because it was reported to be the second best approach.
For the BCA, that is used to generate the "co-occurence probabilities", parameters $\alpha$ (which probability fraction is retained on a node) and $\epsilon$ (minimum value of probability to be distributed, values below being discarded) were chosen identical to the number Cochez et al. chose ($\alpha = 0.1$ and $\epsilon = 0.00001$).
The output co-occurence matrix was then put into *GloVe* using the same parameters as above.

[6] see `https://github.com/dwslab/jrdf2vec`
[7] see `https://radimrehurek.com/gensim/`
[8] see `https://nlp.stanford.edu/projects/glove/`
[9] In order to add another level of consistency between the approaches, the dimensions were also fixed to 200 in all of the other embedding generation approaches.
[10] available under `https://github.com/stanfordnlp/GloVe`
[11] `https://github.com/miselico/globalRDFEmbeddingsISWC`

Six embedding sets were therefore generated in total: two for each of the three corpora.

*Matching Process* Only the label and the entity type (class, datatype, property, object property, or individual) are considered. The entity types are used as a filter to only be matched against each other so that a homogeneous alignment is created, which proved to be a valuable heuristic in development. Matches are mainly determined based on the entity label. In the first step of the pipeline, simple matches are detected by a string matcher assuming n:m arity. Following steps try to apply increasingly less specific background knowledge in the form of embeddings trained on respectively less specific corpora, assuming only 1:1 arity (by ignoring entities already appearing in predicted matches). The specificity was assumed from the vocabulary size of a corpus. Per corpus, *Word2Vec/RDF2Vec* were applied before *GloVe/KGloVe* embeddings[12]. So the embedding sets were applied in the order *FIBO-RDF2Vec*, *FIBO-KGloVe*, *SAP Glossary-Word2Vec*, *SAP Glossary-Word2Vec*, *SAP Term-Word2Vec*, *SAP Term-GloVe*.

*Implementation* The system has been implemented and packaged with the *Matching and Evaluation Toolkit* (MELT), a framework for matcher development, tuning, evaluation, and packaging [4, 10]. As the matcher heavily depends on the python environment, the ML server module [5] of MELT has been forked to wrap additional python code. Eventually, the system was packaged with the framework. MELT greatly facilitated matcher development and also allowed for an easy inclusion of correspondence-level expanations.

## 2 Results

### 2.1 Anatomy

For this track, *DESKMatcher* was barely able to exceed the *StringEquiv* baseline and heavily underperformed on Precision and in turn $F_1$. Because the knowledge to train the embeddings was not taken from the same domain, these results are not surprising.

### 2.2 Conference

The Recall of 0.5 was rather below average compared to other matching systems, whereas Precision and $F_1$ were far below that of the others. An overlap between the Conference vocabulary present in the track and Business vocabulary from the background knowledge might have been expected, which in turn would have caused *DESKMatcher* to perform better.

[12] The decision whether to apply *Word2Vec/RDF2Vec* or *GloVe/KGloVe* embeddings first was taken arbitrarily. An improvement would be to investigate which embedding approaches actually are most suited for matching tasks.

### 2.3 Knowledge Graph

*DESKMatcher* was able to perform all test cases of the knowledge graph track [6]. In order to increase the performance, the embeddings are not used for instance matching. With an $F_1$ of 0.81, the matching system could outperform several systems on this track such as all 2020 *LogMap* [7] matching systems. Yet the F-score is still close to the `baselineLabel` matcher and below the `baselineAltLabel` matcher.

## 3 General Comments

### 3.1 Comments on the results (strength and weaknesses)

This system uses very specific domain knowledge from the financial services and business domains, which are not exactly covered by any of the tracks. Therefore, it was expected, that it should not be able to perform well. Even though expectations were set low, the results appear to be even worse. The system's strength lies in it being able to improve recall, which causes its greatest weakness: bad precision that in turn leads to bad $F_1$.

### 3.2 Discussions on the way to improve the proposed system

The greatest weakpoint of bad precision needs to be removed. Possible solutions would be a more strict linking process. A very greedy linking approach was chosen, to be able to find any matches at all. Additionally, the embedding sets can be pre-evaluated in a different way and discarded or used accordingly; using multiple embedding sets for one corpus did not show any positive results in the datasets evaluated here.

## 4 Conclusions

In this paper, we presented the *DESKMatcher*, a matching system for the financial services domain. The inner workings of the systems have been explained and the performance numbers in the 2020 campaign of the OAEI have been discussed. The system did not perform competitively in the campaign due to low vocabulary overlap in the datasets that have been used. We strive to improve the system in the future.

# Bibliography


1. Berkhin, P.: Bookmark-coloring algorithm for personalized PageRank computing. Internet Mathematics 3(1), 41–62 (2006)
2. Cochez, M., Ristoski, P., Ponzetto, S.P., Paulheim, H.: Global RDF Vector Space Embeddings. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) The Semantic Web – ISWC 2017, vol. 10587, pp. 190–207. Springer International Publishing, Cham (2017)
3. EDM Council: About FIBO. `https://edmcouncil.org/general/custom.asp?page=aboutfiboreview` (2019), (accessed 2020-09-02)
4. Hertling, S., Portisch, J., Paulheim, H.: MELT - Matching EvaLuation Toolkit. In: Semantics 2019 SEM2019 Proceedings. Karlsruhe (2019, to appear)
5. Hertling, S., Portisch, J., Paulheim, H.: Supervised ontology and instance matching with MELT. In: OM@ISWC 2020 (2020), to appear
6. Hofmann, A., Perchani, S., Portisch, J., Hertling, S., Paulheim, H.: Dbkwik: Towards knowledge graph creation from thousands of wikis. In: Nikitina, N., Song, D., Fokoue, A., Haase, P. (eds.) Proceedings of the ISWC 2017 Posters & Demonstrations and Industry Tracks co-located with 16th International Semantic Web Conference (ISWC 2017), Vienna, Austria, October 23rd - to - 25th, 2017. CEUR Workshop Proceedings, vol. 1963. CEUR-WS.org (2017), `http://ceur-ws.org/Vol-1963/paper540.pdf`
7. Jiménez-Ruiz, E.: Logmap family participation in the oaei 2020. OM@ISWC 2020 (2020), to appear
8. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient Estimation of Word Representations in Vector Space (Jan 2013)
9. Pennington, J., Socher, R., Manning, C.: Glove: Global Vectors for Word Representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543. Association for Computational Linguistics, Doha, Qatar (2014)
10. Portisch, J., Hertling, S., Paulheim, H.: Visual analysis of ontology matching results with the melt dashboard. In: The Semantic Web: ESWC 2020 Satellite Events (2020)
11. Portisch, J., Hladik, M., Paulheim, H.: Evaluating Ontology Matchers on Real-World Financial Services Data Models p. 5 (2019)
12. Portisch, J., Hladik, M., Paulheim, H.: Rdf2vec light - A lightweight approach for knowledge graph embeddings. CoRR abs/2009.07659 (2020), `https://arxiv.org/abs/2009.07659`
13. Portisch, J., Paulheim, H.: ALOD2Vec Matcher. OM@ISWC. CEUR Workshop Proceedings (vol. 2288), pp. 132–137 (2018)
14. Řehůřek, R., Sojka, P.: Software framework for topic modelling with large corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50. ELRA, Valletta, Malta (May 2010)
15. Ristoski, P., Paulheim, H.: RDF2Vec: RDF Graph Embeddings for Data Mining. In: Groth, P., Simperl, E., Gray, A., Sabou, M., Krötzsch, M., Lecue, F., Flöck, F., Gil, Y. (eds.) The Semantic Web – ISWC 2016, vol. 9981, pp. 498–514. Springer International Publishing, Cham (2016)
16. Sheikh, I., Illina, I., Fohr, D., Linares, G.: Document Level Semantic Context for Retrieving OOV Proper Names. In: 2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 6050–6054. Proceeding of IEEE ICASSP 2016, IEEE, Shanghai, China (Mar 2016)