

Multi-Concept Alignment and Evaluation

Shenghui Wang^{1,2}, Antoine Isaac^{1,2,3}, Lourens van der Meij^{1,2}, and Stefan Schlobach¹

¹ Vrije Universiteit Amsterdam

² Koninklijke Bibliotheek, Den Haag

³ Max Planck Institute for Psycholinguistics, Nijmegen
{swang,aisaac,lourens,schlobach}@few.vu.nl

Abstract. In this paper we discuss a book annotation translation application scenario that requires multi-concept alignment – where one set of concepts is aligned to another set. Using books annotated by concepts from two vocabularies which are to be aligned, we explore two statistically-grounded measures (Jaccard and LSA) to build conversion rules which aggregate similar concepts. Different ways of learning and deploying the multi-concept alignment are evaluated, which enables us to assess the usefulness of the approach for this scenario. This usefulness is low at the moment, but the experiment has given us the opportunity to learn some important lessons.

1 Introduction

The ontology alignment problem is crucial for many applications. Many methods and tools have been proposed to assist this task by proposing candidate correspondences between two ontologies. Yet the application field is largely uncharted, and some specific problems remain untouched by the community.

One such problem is multi-concept alignment.⁴ In these, a given correspondence⁵ can involve more than one ontological entity on either side. These entities are possibly combined in expressions – *e.g.* using the boolean operators AND and OR – as in `o1:FruitsAndVegetables` → (`o2:Fruits` OR `o2:Vegetables`).

However, only a few ontology alignment tools and methods address the production and exploitation of complex multi-concept alignment [3]. Similar to the difficulties in finding complex matches between database schemas [5], dealing with multi-concept alignment is difficult:

- Finding and using multi-concept alignments have to deal with much more complex search space. If two ontologies have u and v concepts, respectively, there are $u \times v$ potential 1:1 correspondences. However, there will be a combinatorial explosion of the number of possible $m : n$ alignments.

⁴ This is sometimes referred to as multiple, complex, or a combination of one-to-many (1:n), many-to-one (m:1), many-to-many (m:n) alignment [3].

⁵ Here, an alignment between two ontologies is understood as a set of correspondences – mapping links – between entities – classes, properties or expressions – coming from these ontologies.

- It is unclear how to compare the similarity between two sets of concepts, as the semantic relations often cannot be stated explicitly as in [2].
- Alignment frameworks and formats do not say much about the semantics of multi-concept alignment, delegating the problem – and, to an extent, rightly so – to specific ontology languages and/or alignment applications. A typical example is the semantic link between several simple correspondences involving the same entities and a single correspondence involving complex expressions.⁶

Also, in practice, most applications in the ontology alignment community currently focus on rather abstract cases, such as ontology merging or bridging. This requires finding the one most similar entity in the target ontology to the one considered in the source ontology. Accordingly, the evaluation of alignments has relied on purely intellectual assessment of pairs of concepts.

In this paper we will give the applicative motivation for multi-concept alignment — an annotation translation case in the National Library of the Netherlands.⁷ As instances – books – are described or *annotated* by concepts from the two vocabularies to align, our focus is on using statistical methods exploiting co-occurrence of concepts in the annotation of the same book [6], such as the Jaccard similarity measure. Here, we elaborate on this approach from a multi-concept alignment perspective: we take two statistical similarity measures – Jaccard and Latent Semantic Analysis (LSA) – and we test different aggregation methods – clustering and ranking – to build multi-concept alignments with them.

Our goal is to assess the usefulness of these methods for our application scenario. We explore strategies with different levels of accuracy for applying the obtained alignment to our data set. We also train these methods on different sets of annotated objects. The questions we want to answer are the following:

- Do these combinations of statistically-grounded measures and aggregation techniques perform well enough for our scenario?
- Can we improve the performance by carefully selecting the training sets for some of the techniques used?
- Is there a benefit using a sophisticated similarity measure (LSA) for instance-based multi-concept alignment?

In section 2, we present our application scenario – book annotation translation – and explain why it requires multi-concept alignment. Section 3 explains the way we use the dually annotated books to compute similarity measures that are aggregated to obtain multi-concept alignments. In section 4 we detail our experimental setting, focusing on the different strategies that we apply to translate book annotations using the rules contained in the alignments, and on how we

⁶ As in $(o2:Fruits \rightarrow o1:FruitsAndVegetables) \sqcap (o1:Vegetables \rightarrow o1:FruitsAndVegetables)$ and $o1:FruitsAndVegetables \rightarrow (o2:Fruits \text{ OR } o2:Vegetables)$.

⁷ Different from classifying documents in hierarchical categories, in our task, a set of concepts should be aligned to another set of concepts, while both sets represent the same or very similar semantics.

evaluate the results of this process. Section 5 presents the results for the different options tested: similarity measures, training sets, aggregation and rule firing strategies. In sections 6 and 7 we point out some lessons learned and conclude on the perspectives of this research.

2 Problem description

The library context The National Library of the Netherlands (KB)⁸ maintains a large number of collections, among them the *Deposit Collection* containing all the Dutch printed publications (one million items), and the *Scientific Collection* with about 1.4 million books mainly about the history, language and culture of the Netherlands.

Each collection is described according to its own indexing system. On the one hand, the Scientific Collection is mainly described using the GTT, a huge vocabulary containing 35000 general terms ranging from *Wolkenkrabbers* (Skyscrapers) to *Verzorging* (Care). On the other hand, the books contained in the Deposit Collection are mainly indexed against the *Brinkman thesaurus*, containing a large set of headings (more than 5,000) that are expected to be global subject of a book. Both thesauri have similar coverage but differ in granularity, and provide the usual lexical and semantic information found in thesauri: broader and related concept, synonyms and notes.

The co-existence of these different annotation systems, even if historically and practically justified, is not satisfactory from the interoperability point of view. KB is therefore investigating ways to combine the two thesauri, trying to enhance integration while retaining compatibility with the legacy data of both systems. For this reason, mapping concepts between GTT and Brinkman are crucially needed.

Finally, it is important to mention that around 250,000 books are common in both depot and scientific collections, and have therefore been manually annotated with both GTT and Brinkman vocabularies. This allows us to investigate the co-occurrence information in the dually annotated dataset in order to find the semantic alignments between concepts from these two thesauri.

A annotation translation scenario The application scenario in this paper is that one thesaurus (*e.g.* GTT) would be dropped. In such a case, a huge volume of legacy data would have to be made compatible with the indexing system that remains (Brinkman). This requires converting the GTT annotations into equivalent Brinkman annotations.

A first approach would be to find, for each concept from GTT, the one in Brinkman which is the semantically closest. Each time this GTT concept would appear in a book description, the corresponding Brinkman concept could be added. Yet such a one-to-one conversion is not satisfactory. First, and most intuitively, one observes that the two vocabularies do not have the same granularity.

⁸ <http://www.kb.nl>

Sometimes for a concept in a vocabulary it is impossible to find a corresponding concept, with the same meaning, in the other vocabulary. This is the case for instance for the Brinkman **gassen** ; **mechanica** (gas mechanics) which has no equivalent in GTT, while the latter includes both **Gassen** and **Mechanica**. This suggests the necessity to introduce alignment links involving several concepts from the same vocabulary. This is confirmed by a second aspect which is more guided by the vocabulary application itself. The GTT annotation process makes use of *post-coordination*: multiple concepts found in a same annotation shall not be considered independent but rather as facets of a more complex virtual subject. With GTT, if a book is annotated by **historische geografie** and **Nederland** you expect the book to be about a more complex “historical geography of the Netherlands” subject instead of being about these two individual subjects in a disconnected manner. Ideally, a conversion algorithm would therefore have to exploit alignments that involve concepts that are combined together. In this way, implicit complex subjects would be properly detected and converted when they occur in an annotation.

3 Multi-concept alignment generation

Closely related concepts form a virtual conceptual entity, and the alignment is generated in terms of these virtual entities. An intuitive way of generating such a virtual entity is to group similar concepts together.

3.1 Instance-based similarity measures

We base our concept aggregation on the similarity between concepts. In [6], some similarity measures we have investigated generate 1:1 mappings between concepts based on their co-occurrence in annotations of books. In this paper we further investigate two measures.

Jaccard similarity The Jaccard similarity coefficient is a simple measure for similarity of sets:

$$J(A, B) = \frac{|A^i \cap B^i|}{|A^i \cup B^i|}$$

where A^i is the set of instances of concept A , in our case, the books which are annotated by A . If there is a perfect correlation between two concepts A and B , the measure will have a value of 1, if there is no co-occurrence, the measure is 0.

Latent Semantic Analysis (LSA) [7] was used to analyse the concept-book co-occurrence matrix and calculate the similarity between concepts. LSA is a statistical technique developed for extracting and representing the similarity between words and between documents by analysis of large bodies of text. In our context we expect the method to provide a measure for the correlation

between concepts in annotation (corrected for insufficient data), as well as a way to distinguish the relevant correspondences between such concepts.

The occurrence of each concept in the annotation of each book is first counted and stored in a concept-by-book matrix $X_{c \times b}$. By using the singular value decomposition (SVD), the matrix $X_{c \times b}$ is decomposed as

$$X_{c \times b} = C_{c \times r} S_{r \times r} B_{r \times b}^T,$$

where c is the number of concepts, b is the number of books, $C_{c \times r}$ describes the original concepts as vectors in a space of r derived orthogonal factor values, $B_{r \times b}^T$ describes the original book annotations in the same way, and $S_{r \times r}$ is a diagonal matrix containing scaling values, which are all positive and ordered in decreasing magnitude. Using only the k largest eigenvalues, a reduced matrix is reconstructed as

$$X_{c \times b} \approx \hat{X}_{c \times b} = C_{c \times k} S_{k \times k} B_{k \times b}^T,$$

which is closest in the least squares sense to $X_{c \times b}$. The aim of this *dimension reduction* is to capture the most important structure but reduce noise and variability in concept usage. We used the percentage of accumulated singular values to determine k . In our case, we kept 80% accumulation.

The product

$$D_{c \times c} = \hat{X}_{c \times k} \hat{X}_{c \times k}^T = (C_{c \times k} S_{k \times k})(C_{c \times k} S_{k \times k})^T$$

gives the paired similarity matrix between concepts [1].

Both methods provide measures of similarity between pairs of concepts. We will see that different similarity measures group concepts in different ways and their performance in our evaluation also vary.

3.2 Concept aggregation

Here we introduce two ways of aggregating concepts using the similarity measures calculated above.

Grouping concepts based on 1:1 mappings For a specific concept C_0 , a list of concepts which are the top k ranked in similarity is easily generated, i.e.,

$$C_0 \rightarrow (C_1, C_2, \dots, C_k).$$

This list is not limited to contain the concepts from the different thesauri only, instead, it contains concepts from both thesauri, ranked by their similarity to concept C_0 .

Once a threshold k is chosen,⁹ the top k concepts together with concept C_0 are expected to form a closely related conceptual entity. Dividing this set of $k + 1$ concepts into concepts from one thesaurus and the other, the two sets of

⁹ In our experiments, we chose $k = 10$.

concepts from both thesauri are used to define an n to m mapping between both thesauri. That is, if

$$G_0 \rightarrow (B_1, G_1, \dots, B_m, G_n),$$

where $n + m = k$, then a mapping rule

$$(G_0, G_1, \dots, G_n) \rightarrow (B_1, B_2, \dots, B_m)$$

is generated.

Partitioning concepts based on clustering One similarity-based clustering technique [4] was used to partition the concepts into clusters. If one cluster contains k concepts

$$(B_1, G_1, \dots, B_m, G_n),$$

where $n + m = k$, then a mapping rule

$$(G_1, \dots, G_n) \rightarrow (B_1, B_2, \dots, B_m)$$

is generated.

The clustering technique takes the similarity between all concepts into account, therefore the generated clusters partition the concepts in a global manner. In this way, the generated $n : m$ mappings are expected to reproduce the general correlation between concepts from both thesauri.

4 Evaluation

We use the book annotation translation scenario to evaluate the generated $n : m$ alignment. The complete set of dually annotated books was divided into two parts: 2/3 of books was used for training, and the rest 1/3 of books as the testing data. Using different sampling methods, we have two training data sets:

Train_{random}: randomly selected books (5245 books and 7391 concepts)

Train_{rich}: books with at least a total of 8 annotations from both thesauri (5288 books and 10382 concepts)

4.1 Evaluation method

From the training data, we learn the alignment rules specifying which set of GTT concepts should be aligned to which set of Brinkman concepts, that is,

$$R : G_r \rightarrow B_r,$$

where G_r is a set of GTT concepts and B_r is the corresponding set of Brinkman concepts. In the testing dataset, each book has its GTT and Brinkman annotation, i.e. G_t and B_t . The GTT annotation was used to fire rules. This results in a generated set of Brinkman concepts B_r' . By comparing B_t and B_r' , we can evaluate the precision and recall of the learned alignment rules.

We now specify how to apply the rules to the GTT annotation and how to define precision and recall. Given an alignment rule $R : G_r \rightarrow B_r$ and a book out of the testing data with the annotations G_t and B_t , we define four different strategies – later denoted by *FireIf* – for firing rules:

- 1 $G_t = G_r$
 - 2 $G_t \supseteq G_r$
 - 3 $G_t \subseteq G_r$
- ALL* Fire in all above three cases.

The different generated Brinkman concepts were distinguished by a subscript i : B_{r_i} . We consider a book to be *matched* if its real Brinkman annotation and the generated set of Brinkman concepts overlap i.e. $B_t \cap B_{r_i} \neq \emptyset$, $i \in \{1, 2, 3, ALL\}$.

4.2 Precision and Recall

Precision and recall are calculated at two levels. At the book level, we measure the performance in terms of the fired books, which were fired by at least one rule. We define the precision as the fraction of the books in the testing data set that actually match their real Brinkman annotations, i.e.,

$$P_b = \frac{\#books_matched}{\#books_fired},$$

and the recall as how many books in the whole testing set are matched, i.e.,

$$R_b = \frac{\#books_matched}{\#books_testing}.$$

where $\#books_matched$ is the number of books whose real Brinkman annotation overlap the generated set and $\#books_testing$ is the number of books in the testing data.

At the annotation level, we measure how well the generated set of Brinkman concepts match the real annotation, i.e.,

$$P_a = \frac{\sum \frac{\#good_found}{|B_{r_i}|}}{\#books_fired}, \quad R_a = \frac{\sum \frac{\#good_found}{|B_t|}}{\#books_testing},$$

where $\#good_found$ is the number of the real Brinkman concepts which are found in the generated set.

5 Results

Table 1 gives an overview of the relation between generated rules, training sets and methods. The size of GTT concepts in these rules is generally 2 or 3 times bigger than that of Brinkman concepts, which is consistent with the way of

Similarity	Methods	Training Set	#Rule	#GTT	#Brinkman
Jaccard	Ranking	random	5669	6.4	4.6
		rich	8334	8.6	4.2
	Clustering	random	246	15.1	5.8
		rich	242	84.2	14.4
LSA	Ranking	random	6916	6.2	4.2
		rich	7117	7.6	3.5
	Clustering	random	747	3.2	1.8
		rich	883	8.2	2.9

Table 1. Generated rules using different training sets and methods

Similarity	Methods	Training Set	P_b	R_b	P_a	R_a
Jaccard	Ranking	random	63.77%	12.26%	12.45%	10.26%
		rich	43.41%	12.43%	5.36%	9.83%
	Clustering	random	26.87%	3.59%	25.17%	2.42%
		rich	5.37%	0.80%	4.16%	0.53%
LSA	Ranking	random	67.55%	17.94%	6.60%	15.16%
		rich	62.51%	19.54%	8.10%	16.41%
	Clustering	random	39.68%	9.19%	22.06%	6.76%
		rich	33.03%	8.01%	10.65%	6.24%

Table 2. Performance overview of different methods

using GTT concepts for annotation, *i.e.*, *post-coordination*. Several GTT concepts should be combined in order to correspond to a single Brinkman concept. One GTT concept in different combinations could well be aligned to different Brinkman concepts.

Note that, when generating rules from the ranked lists, the top 10 most similar concepts were grouped with a target concept, therefore, the sum of GTT and Brinkman concepts is around 11. The ranked lists of different concepts may contain the same group of concepts, so, in the end, the number of the rules is different and less than the total number of concepts. The clustering method creates partitions of concepts and each cluster generates one rule, so the generated rules are much less than those from the ranking method.

In Table 2, we compare the performance of different methods using different training sets, with the firing type *ALL*. One message from the table is that the arbitrary choice of using richly annotated books as training data, in general, does not bring benefits. Instead, as it is biased towards the richly annotated books, it gets unnecessary information involved which deteriorates the general performance in the normal cases.

However, using *Train_{rich}* has different effects on two similarity measures in the case of ranking. It increases the performance when using LSA for ranking but decreases for Jaccard measure. This is because the Jaccard measure gets confused by the sophisticated data while grouping concepts from simple ranked list is also not capable of coping with such data. Instead, LSA makes statistically

FireIF type	Method	C_r	P_b	R_b	P_a	R_a
1	Jaccard	47.56%	62.77%	0.65%	60.46%	0.55%
	LSA	21.95%	47.00%	0.57%	37.32%	0.44%
2	Jaccard	58.94%	50.43%	3.39%	47.76%	2.27%
	LSA	46.05%	36.95%	3.22%	29.98%	1.99%
3	Jaccard	88.21%	11.08%	0.85%	10.14%	0.69%
	LSA	82.20%	41.77%	6.53%	18.82%	5.21%
ALL	Jaccard	97.15%	26.87%	3.59%	25.17%	2.42%
	LSA	92.90%	39.68%	9.19%	22.06%	6.76%

Table 3. Comparison between Jaccard and LSA in clustering method, where C_r is the ratio of the fired rules over the total number of rules for the given technique.

valid corrections which compensates for the simple strategy of grouping based on similarity ranking.

5.1 Comparing Jaccard and LSA

By matching type *ALL*, the LSA similarity outperforms the simple Jaccard measure, in terms of the precision and recall of both ranking and clustering methods. Table 3 gives the detailed figures in terms of different rule firing types. On the one hand, using the simple Jaccard measure, more books are fired with the exact and subsume match (type 2). This indicates that Jaccard measure is good at finding explicit similarity from the co-occurrence information.¹⁰ On the other hand, LSA is able to find some potentially similar concepts. Therefore, by slightly sacrificing in the precision of generated annotations, LSA improves significantly the precision and recall at the book level also gives the higher recall at the annotation level.

5.2 Comparing Ranking and Clustering

In Table 2, for type *ALL*, rules based on ranking have better precision and recall than those generated from clustering, except that the rules from the clusters have higher precision in terms of the generated annotations. Table 4 gives the detailed comparison between rules generated from ranking and clustering. Here, the similarity measure is LSA.

The rules generated from the clusters have better performance when fired by type 1 and 2. More than 20% rules were fired by the exact match. This indicates that those clusters matches the real annotations quite well, at least the GTT concepts within clusters are highly correlated and matches the real data. The high precision also guarantees the usefulness of the generated Brinkman annotations. The higher recall and lower precision of the rules generated by ranking is because that more concepts were involved in the ranked list, where more good candidates and noisy data are introduced at the same time.

¹⁰ This is also confirmed in [6].

FireIF type	Method	C_r	P_b	R_b	P_a	R_a
1	Ranking					
	Clustering	21.95%	47.00%	0.57%	37.32%	0.44%
2	Ranking	0.01%	100.00%	0.01%	50.00%	0.01%
	Clustering	46.05%	36.95%	3.22%	29.98%	1.99%
3	Ranking	98.54%	67.55%	17.94%	6.60%	15.16%
	Clustering	82.20%	41.77%	6.53%	18.82%	5.21%
ALL	Ranking	98.54%	67.55%	17.94%	6.60%	15.16%
	Clustering	92.90%	39.68%	9.19%	22.06%	6.76%

Table 4. Comparison between rules generated from ranking and clustering, using LSA measure.

6 Discussion

As shown in Table 1, using simple Jaccard similarity, the average size of clusters is much bigger compared to other methods. Those clusters in fact have a big variation in their size. Due to the existence of big clusters, many books are fired because their GTT annotations are subsumed by the rules (type 3). Because of the quality of the bigger clusters is somehow low, the generated Brinkman annotations are not really good, which decreases the precision and recall. However, if only counting smaller clusters (the size is equal to or less than 11), 80% clusters if using *Train_random* (59% if using *Train_rich*) contains 1.4 GTT and 1.2 Brinkman concept (1.9:1.2 if using *Train_rich*). These small clusters generally contain the concepts which are strongly related, therefore, 1:1 or 2:1 mappings generated by them are expected to be more reliable.

Rule size also matters when generating rules from the ranked list. The choice of k directly affects the size of rules. In our case, the choice of 10 produces rules which involve too many concepts. Therefore, many books are fired because their GTT annotations are subsumed by the rules (type 3) while very few books are fired by exact (type 1) or subsume matches (type 2). It would be interesting to investigate the effect of the choice of k on the precision and recall performance.

Another concern is that one GTT concept could well be mapped to different sets of Brinkman concepts if it is combined with different GTT concepts. However, using the clustering methods, concepts are allocated into only one partition. This limits the number of rules as well as the final performance. We will investigate some probabilistic clustering techniques in order to allow more flexible rules.

7 Conclusion

In this paper we have introduced an application that needs multi-concept alignment. We have presented and evaluated strategies to generate and deploy such alignments, using statistical techniques that are expected to take into account the way concepts are combined in the instance-level annotation data.

At this point we still have to answer the question whether the measures and techniques we have used perform well for our annotation translation scenario. First, the precision of the generated sets of Brinkman concepts (P_a) (see table 2) is not encouraging. Ranging from 4.16% to 25.17%, it prevents from trusting the candidate concepts as a unique source for annotation translation. A scenario where a user would be proposed these candidates and required to assess them to produce valid annotation would be more realistic: having to choose among, for example, 10 Brinkman concepts is already an improvement compared to selecting among all the Brinkman vocabulary. Additionally, table 4 shows that selecting the rule firing strategy can improve the precision. Yet, such a scenario seems to be ruled out by the annotation-level recall (R_a): in the most effective configuration, 16.41% of the correct annotations were found. The book-level recall (R_b) shows that at most 19.54% of the books in the testing set were given at least one good candidate. This means that an annotator should add his own concepts on top of the candidate ones. One single technique is therefore not enough to cope with the annotation translation for an entire collection.

Section 6 has shown that these weaknesses might be compensated by using aggregation techniques that approximate better the way concepts are used for book annotation. The objective there would be to increase the number of the more reliable rules to produce more valid results (type 1 and 2). Also, different strategies used here can be combined so as to obtain better performance. These two options have to be explored further.

Also, an immediate way to improve the results without even changing the methods could be to revisit the evaluation itself. Our evaluation technique gives a first and cheap assessment, which is likely to remain generally valid. Yet it is sensitive to *indexing variation*, the phenomenon that renders the fact that several annotators annotating a same book (or a same annotator annotating it at different time) will select slightly different concepts. Some manual application-specific evaluation shall be performed to assess the influence of this phenomenon and eventually compensate for its bias.

References

1. Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of The American Society for Information Science*, 41(6):391–407, 1990.
2. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: Discovering complex semantic matches between database schemas. In *Proceedings of the ACM International Conference on Management of Data (SIGMOD)*, pages 383–394, 2004.
3. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag, 2007.
4. Brendan J. Frey and Delbert Dueck. Clustering by passing messages between data points. *Science*, February 16 2007.
5. B. He, K. C.-C. Chang, and J. Han. Discovering complex matchings across web query interfaces: A correlation mining approach. 2004.
6. Antoine Isaac, Lourens van der Meij, Stefan Schlobach, and Shenghui Wang. An empirical study of instance-based ontology matching. In *Proceedings of the 6th International Semantic Web Conference*, 2007. To appear.

7. T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25:259–284, 1998.