Supporting Collaborative Modeling via Natural Language Processing

Fatma Başak Aydemir[1] and Fabiano Dalpiaz[2]

[1] Boğaziçi University, Turkey basak.aydemir@boun.edu.tr
[2] Utrecht University, The Netherlands f.dalpiaz@uu.nl

Abstract. Engineering large-scale systems requires the collaboration among experts who use different modeling languages and create multiple models. Due to their independent creation and evolution, these models may exhibit discrepancies in terms of the domain concepts they represent. To help re-align the models without an explicit synchronization, we propose a technique that provides the modelers with suggested concepts that they may be interested in adding to their own models. The approach is modeling-language agnostic since it processes only the text in the models, such as the labels of elements and relationships. In this paper, we focus on determining the similarity of compound nouns, which are frequently used in conceptual models. We propose two algorithms, that make use of word embeddings and domain models, respectively. We report an early validation that assesses the effectiveness of our similarity algorithms against state-of-the-art machine learning algorithms with respect to human judgment.

Keywords: collaborative modeling · conceptual modeling · natural language processing · semantic similarity

1 Introduction

The systems we build are growing in size and complexity [25]; many examples are before our eyes such as intelligent transportation systems, healthcare infrastructures, and smart grids. Due to the complex interactions between systems and their subsystems [30], we need to analyze cross-cutting system concerns such as performance, security, safety, privacy, and fairness, through approaches like aspect-oriented design [7].

Specific expertise is necessary to study these different aspects, thereby demanding collaboration among multiple experts. Since physically bringing these experts to the same location is quite costly and even not possible due to unforeseen situations such as a pandemic, the collaboration among these individuals increasingly relies on mediated, asynchronous interaction over the Internet thanks to the rise of digital transformation.

One of the challenges in this setting is to create consistent models. Modeling techniques can ensure consistency via, e.g., aspect weaving [14,34], when a common metamodel exists. As an alternative, recent work [24] has proposed near real-time modeling frameworks for enabling collaboration on the same model and for resolving conflicts.

We take a complementary standpoint. Instead of integrating the models into a consistent supermodel, we propose algorithms for achieving *model alignment*; two models are aligned when they capture similar if not the same domain concepts and relations.

If the models are aligned, it means that the experts have analyzed the same part of the domain, and can therefore conduct trade-off analyses between the various aspects.

We assist model alignment by *suggesting concepts* that the modelers may want to include in their models. We suggest concepts that are missing in a model but are captured in other models in the same project. To identify candidate concepts, we automatically search for element and relationship labels with identical or similar noun phrases.

Identifying matching or missing concepts is not trivial. On one hand, our experience in system specification [23] has shown that modelers (and designers in general) often use *compound nouns* to describe a domain, e.g., 'company car', 'car engine', 'car rental'. On the other hand, interpreting the meaning of compound nouns is renown to be difficult [20,22]. As a consequence, finding synonym compound words is all but simple.

Our work takes a lightweight approach to matching terms, in which we do not assume the definition of a shared meta-model [24], the existence of a domain ontology [11], or knowledge about the semantics of the model elements [21].

In this paper, we make three specific contributions toward the alignment of multiple models without requiring knowledge of or aligning the meta-models:

1. We propose the Concept Suggester service that relies on natural language processing (NLP) heuristics to propose additional concepts to the modelers; see Sec. 2. This paper extends the vision of the service sketched in previous work [2].
2. We devise two algorithms, that use different approaches, for computing the similarity of two-word compound nouns (Sec. 3).
3. We report on an experiment that assesses the performance of our heuristics against off-the-shelf machine learning algorithms (Sec. 4).

After discussing related work in Sec. 5, we conclude and present future work in Sec. 6.

2 The Concept Suggester Service

The *Concept Suggester* service supports the asynchronous collaboration among two or more modelers by analyzing the changes they make in their own models, and by recommending ("suggesting") which domain concepts they may be interested to include in their models by analyzing the domain concepts that are represented in other models.

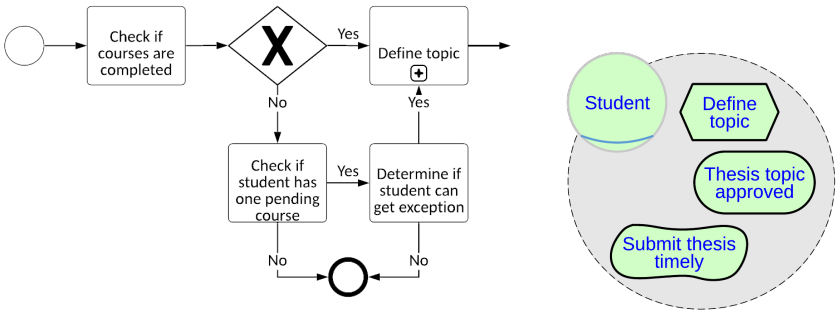
2.1 Motivation and System Overview

The concept suggester service was conceived in the context of the PACAS research project[1] regarding the decision making processes in the air traffic management domain. In this setting, multiple organizations (such as governments, airports, and airlines) investigate possible solutions from different perspectives through models that are built using various modeling languages. Currently, experts from various organizations, working in different time-zones and locations, build their models during face-to-face workshops. Besides costs and carbon footprint, this type of workshops become impossible in time of a pandemic. Note that this situation applies to other kinds of information systems in which different experts collaborate to model the system from different perspectives.

[1] https://www.sesarju.eu/projects/pacas

This setting poses some challenges. How to ensure that the individual models, created independently, analyze the same domain by representing the same domain concepts (*C1*)? Also, in case of competing collaboration (e.g., airline companies), the modelers may want to not disclose sensitive information while allowing alignment (*C2*). Finally, how to ensure the use of unified terminology (*C3*)? This challenge stems from the different backgrounds of experts, which may result in different labels for the same concepts.

Our service focuses on overcoming these challenges. Regarding *C1*, it keeps track of the concepts modelled in each model, without looking into the actual meaning of the models. When domain concepts are identified that miss in other models, those concepts are suggested to increase model completeness. *C2* is addressed by not sharing with other modelers the actual content of the models and the relationships between concepts. Regarding *C3*, a domain ontology is consulted when making suggestions to encourage including standard terms rather than the jargon of the individual modelers. Note that, by not requiring a shared meta-model, the experts are free to use their modeling languages.



(a) An incomplete BPMN model (b) An incomplete iStar model

Fig. 1: Early models for a thesis management system

Illustration. Fig. 1 presents two incomplete models drawn in the early phases of the design of a thesis management system for a higher education institution. Our service recognizes the concepts modelled during each modeling session, and keeps track of the modelled concepts for each model. It also tracks the missing concepts for each model, and suggest them to the modeler. In this illustrative example the `course` concept appears in the BPMN model (Fig. 1a), but the iStar goal model (Fig. 1b) lacks it. Then, our service suggests this concept to the modeler to support the modeling process.

The Concept Suggester was originally built for the PACAS project to support the collaboration among air traffic management experts working on security, safety, performance, etc. Through its integration in the PACAS collaborative modeling platform[2], we obtained feedback from domain experts that led to the version described here.

Fig. 2 shows a typical interaction between two modelers, mediated by the Concept Suggester. Modeler 1 commits a model m1 that she created. A *commit* operation (O1) denotes a significant change that a modeler aims to share. The model is analyzed (O2)

[2] https://pacas.disi.unitn.it/pacas

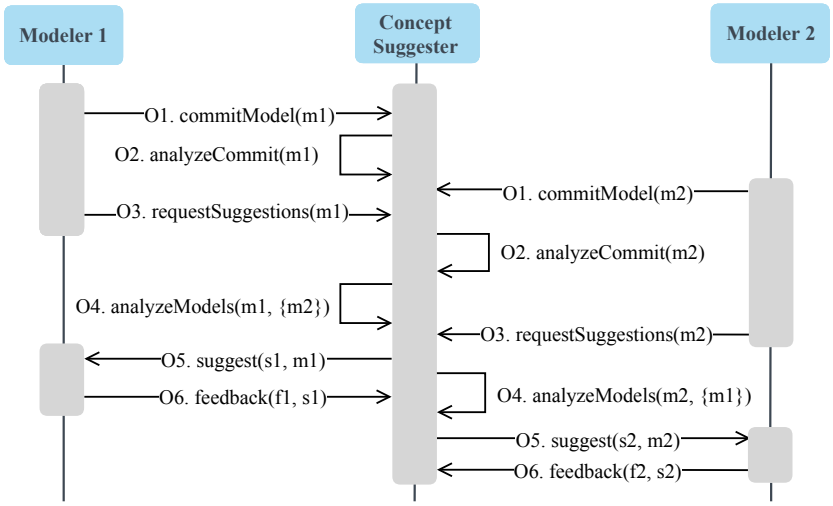


Fig. 2: Interaction between two modelers and the Concept Suggester.

by the service, which identifies the noun phrases in the element labels. Then, Modeler 2 commits her own model m2, which was created independently from m1. Modeler 1 requests suggestions (O3) of concepts to include in her model: the service, after analyzing the other model m2, looks for concepts in m2 that do not appear in m1, which are suggested (O5) to Modeler 1. This actor provides feedback (O6) on the usefulness of the suggested concepts. A similar cycle takes place between Modeler 2 and the service.

2.2 NLP heuristics for term-concept matching

Operations O2 and O4 use NLP algorithms that process the element labels in a committed model (O2), and that identify domain concepts that the committed model does not include but that are included in other models of the project (O4). Below, *terms* are the noun phrases in model elements, while *concepts* are the elements in a domain model. The BPMN diagram of Fig. 3 shows the NLP that takes place between O1 and O4.

The process starts when Model 1 is committed (O1). Noun phrases are extracted from the committed model (O2), resulting in the set of terms from that model. Then, when the modeler requests suggestions for concepts to include in her model, the extracted terms are compared against project terms from other models (O4a), with the goal of identifying project terms that do not appear in the processed model. Then (O4b), these missing terms are matched against the domain model, filtering out those terms that are not domain-specific, and resulting in the candidate concepts to recommend.

Matching heuristics. Many algorithms exist to determine whether two terms or concepts do match, ranging from exact lexical match to more advanced metrics like semantic similarity. Some examples: *i. Exact string match*: two terms/concepts match only if they are the same string; *ii. Substring match*: one term/concept is a substring or a superstring of another; *iii. Similarity*: the semantic similarity between the terms/concepts is

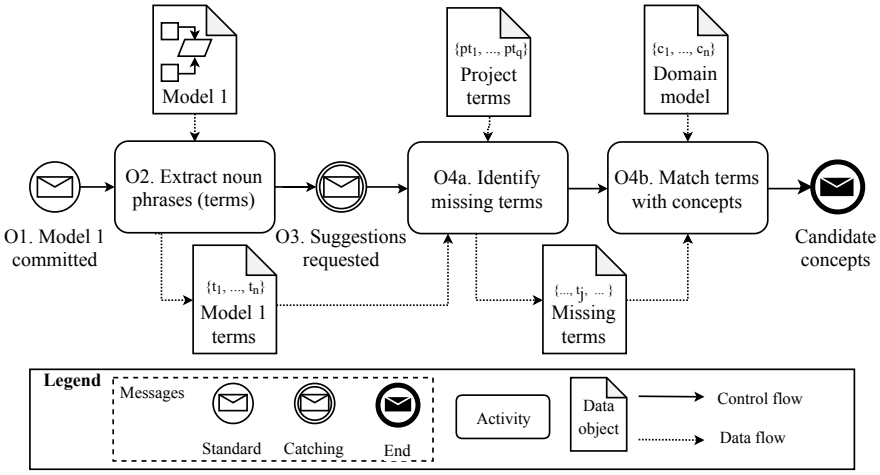


Fig. 3: From model commit to candidate concepts to suggest.

above a certain threshold, e.g., computed by counting the shared *is-a* relationships in an ontology [17]; *iv. Relatedness*: like similarity, but based on the number of relationships—of any type, *is-a* or otherwise—that two concepts share in an ontology [17].

Exact string match provides the most restricted set of suggestions since it does not leave room for concept exploration. It is therefore particularly useful towards the end of the overall modeling process to ensure model alignment. The other heuristics provide suggestions in a broader spectrum, and may be useful earlier in the modeling process, for they suggest divergence, triggering creativity and increasing the scope of the model.

Suggesting concepts. When missing concepts are found by O4, and we possess a domain model that has a graph structure (e.g., a taxonomy or an ontology), O5 can optionally navigate the graph structure to suggest additional concepts. Possible strategies include: *i. Parent:* suggestions at a higher level of abstraction by recommending the parent of the matching concept in the domain model; *ii. Children:* more detailed recommendations by suggesting the child nodes of a matching concept; and *iii. Sibling*: recommending the siblings of a matching concept to foster creativity and extend domain coverage.

For example, using *parent*, we may recommend ‘aerodrome operations’ instead of ‘de-icing’, for ‘de-icing’ specializes ‘aerodrome operations’ in the Air Traffic Management Information Management Reference Model (AIRM, http://airm.aero). Using *children*, if ‘air traffic operations’ is a match, then possible suggestions could be ‘aerodrome operations’, ‘ATM service delivery management’, ‘airspace user operations’, and other child nodes in AIRM. Employing *sibling*, if ‘aerodrome operations’ is a match, a sibling like ‘airspace user operations’ could be suggested.

Filtering the suggestions. Depending on the number of missing concepts and how many matching concepts exist in the domain model, the number of suggestions might grow rapidly. Our experience with domain experts in PACAS led us to devising some strategies: *i. Fixed number:* a maximum number of suggested concepts is set (our domain experts suggested five to ten suggestions at a time); *ii. User feedback:* when a modeler

expresses a suggestion is irrelevant (O6), the service blacklists that concept and related ones; *iii. Frequency:* the most recurring concepts that are present in other models are suggested; *iv. Limiting the matches per missing term:* this filter selects a limited number of matches per missing term, so to support divergence.

3 Similarity for Compound Nouns

In our initial implementation [2], we adopted nouns as units of computation, for which off-the-shelf NLP libraries exist that can measure similarity. When we observed that many labels use compound words, we switched to noun compounds as the unit of computation, and realized that the NLP literature is much weaker on compounds similarity [20,22]. As such, we developed two heuristics for supporting this task.

The first heuristic (Sec. 3.1) combines semantic similarity measures that rely on word2Vec and WordNet and that use general-purpose corpora. The second heuristic (Sec. 3.2), instead, uses a domain glossary as a type of domain model. Both heuristics return a similarity score in the $[0, 1]$ interval.

3.1 Using word embeddings and WordNet

This heuristic measures the similarity between a pair of two-word compounds through a combination of *i.* their lexical similarity with *ii.* the semantic similarity between the words that compose the compounds, determined using word embeddings and WordNet. We use word embeddings because of their good results for short text similarity [18].

Algorithm 1: Similarity of two words.

```
1 Function tws
      Data: wordA, wordB
2     similarity ← word2VecSim(wordA, wordB)
3     if similarity > σ then
4         if (synsets(wordA) ∩ synsets(wordB)) ≠ ∅ then
5             return 1
6     return similarity
```

Alg. 1 calculates the similarity of two words. Line 2 invokes the word2Vec similarity algorithm that relies on word embeddings; word2Vec is implemented in state-of-the-art NLP toolkits like spaCy and NLTK. If the obtained similarity is greater than a given threshold σ, line 4 checks if they are synonyms by intersecting their synonym sets in WordNet. If the intersection is not empty, the algorithm returns 1: full similarity, i.e., synonymy. In all other cases, it returns the similarity value from word2Vec.

Alg. 1 is invoked by Alg. 2 and determines the similarity of a pair of two-word compounds. This second algorithm measures the similarity of each word with the other words in the other compound by calling Alg. 1 for each combination, and combines the results into the overall similarity score using a weighted sum. The values of the similarity threshold σ (Alg. 1), and the weights for the similarity of first words, second words, and cross words γ, δ, and ϵ (Alg. 2) are assigned as 0.6, 0.3, 0.4, and 0.15, respectively,

Algorithm 2: Similarity of two-word compounds.

```
1 Function getCompoundSimilarity
      Data: cwordA =[w1, w2 ], cwordB =[w3, w4 ]
2     return
        γ · tws(w1, w3) + δ · tws(w2, w4) + ϵ · (tws(w1, w4) + tws(w2, w3))
```

for they correlate best with an evaluation by human experts [6] ($r = 0.621$: moderate to strong correlation). Future research is needed to further validate these weights.

3.2 Using domain model matching

The `getSimilarityViaDM` algorithm computes the similarity between two n-word compounds with the help of a domain model. The idea is to denote similarity between two compounds only when both are similar to a given concept in the domain model. Differently from the heuristic in Sec. 3.1, this algorithm is domain specific.

Alg. 3 formalizes this intuition. Given two compound words and a domain model, the similarity score is first set to zero (line 2). Then, a cycle iterates through all concepts in the domain model (lines 3–10), and calculates the similarity between the two words with respect to the domain concept at hand. Lines 4 and 5 calculate the match score between each word and the domain concept (details below). If both match scores are greater than zero (line 7), the similarity between the two words w.r.t. the given concept is computed as the average of the match scores (line 8). If the score is greater than the maximum similarity score between the two compounds computed w.r.t. previously processed domain concepts (line 9), the maximum similarity score is updated (line 10).

Algorithm 3: Similarity score between compound words via a domain model.

```
1  Function getSimilarityViaDM
       Data: wordA, wordB, domainModel
2      simScore ← 0
3      foreach dc ∈ domainModel do
4          matchScoreA ← getMatchScore(wordA, dc)
5          matchScoreB ← getMatchScore(wordB, dc)
6          abScore ← 0
7          if matchScoreA > 0 ∧ matchScoreB > 0 then
8              abScore ← (matchScoreA + matchScoreB) / 2
9              if abScore > simScore then
10                 simScore ← abScore
11     return simScore
```

The function `getMatchScore` calculates the match score between a compound word *w* and a concept *dc* in the domain model. It assigns a full match score (1.0) when *w* and *dc* are the same string, and a slightly lower score (0.75) when *w* is a substring of *dc* (e.g., $w =$'air traffic' and $dc =$'air traffic controller'). Following a similar rationale, it considers the case of a substring of w that excludes the first word: we assign score 0.5 when that corresponds to the domain concept ($w =$'congested air traffic' $dc =$'air

traffic'), and 0.4 when the shortened version of w is a substring of dc. Finally, it assigns even lower scores (0.3 and 0.2) by considering the substring of w that removes the last word; e.g., a score of 0.3 would be assigned with w ='air traffic control' and dc ='traffic control'. The scores for the two substring variants differ because the former case (removing the first word of a compound) corresponds to following bottom-up a specialization relationship, the principle used by similarity metrics based on WordNet.

4 Evaluation

We evaluate the effectiveness of our measures for the similarity of compounds with respect to human judgment; our research question is the following:

RQ Given the similarity measures obtained by the techniques of Sec. 3, which one aligns best with human judgment?

4.1 Experimental setup

We identified four categories to measure the similarity of two-word compound pairs: *Cross*–the first word of a compound is the same as the second word of the other compound in the pair; *First*–the compounds share the same first word; *Second*–the compounds share their second word; *None*–the compounds do not share any word. Next, we randomly picked five instances of these categories from the set of concepts that we have extracted from the website of the Master of Business Intelligence (MBI) program of Utrecht University (archived as *Exp. MBIThesisWebSite* in [3]). The left part of Fig. 4 shows these pairs of compounds and their corresponding categories.

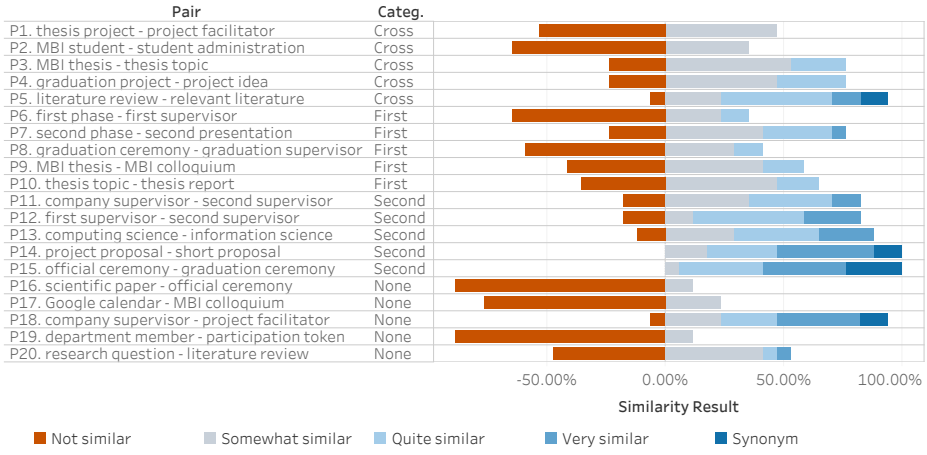


Fig. 4: Similarity tagging of the 20 pairs of compounds.

The second author created a domain model. This was done by first executing a Python script (*Exp. ParseNounChunks* in [3]) that extracts noun chunks from the MBI website. Then, the output was analyzed manually to retain only domain-specific terms

and to identify synonyms among those terms. This manual processing required 1 hour of work, and the output is *Exp. DomainModel* in [3].

Via an online survey (*Exp. Survey* in [3]), we asked humans to assess the similarity between these pairs via a 5-point Likert type scale consisting of the values *not similar*, *somewhat similar*, *quite similar*, *very similar*, and *synonym*. We measured the compounds' similarity using four methods: *i.* word embeddings with WordNet (Sec. 3.1), *ii.* domain model matching (Sec. 3.2), *iii.* spaCy's similarity algorithm that relies on word embeddings [31], and *iv.* an implementation of semantic similarity[3] that trains Google BERT on the STS benchmark [5].

Participants. We sent the survey to the students and academic staff of the MBI program as they are familiar with the domain; we received 17 responses. The participants spent an average of 6 minutes and 55 seconds to fill the survey. Besides evaluating the similarity of pairs using the scale described above, the participants could also indicate the most challenging pairs and provide feedback.

4.2 Results: gold standard

The right part of Fig. 4 and Fig. 5 show the results of the similarity tagging for each of the compounds and grouped by category (cross, first, second, none), respectively. We converted the 5-point Likert scale to a $[0, 1]$ similarity score with 0.25 increments, and assigned the mean of the scores assigned by the taggers as the gold standard score. All the responses are in *Exp. RawData* in [3]. Although our little sample size requires caution in our interpretation, it seems that the highest similarity is achieved when the pairs share the second word. This is in line with the specialization relationship: for example, first supervisor and second supervisor specialize the concept supervisor.

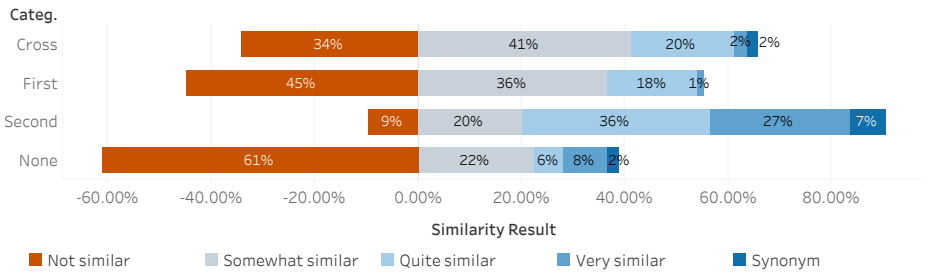


Fig. 5: Similarity tagging, grouped by categories.

8 taggers found the couple company supervisor - project facilitator (P18) among the most difficult pairs to evaluate, followed by four couples highlighted by 6 taggers each: graduation project - project idea (P4), literature review - relevant literature (P5), and project proposal - short proposal (P14). The participants' opinions show how difficult it is to evaluate compounds by focusing on similarity, rather than focusing on relatedness (operationalized via co-occurrence by most NLP toolkits).

[3] https://github.com/AndriyMulyar/semantic-text-similarity

4.3 Results: algorithms

Fig. 6 compares the similarity values by the different heuristics and the gold standard, splitting the results per category. The outputs of the algorithms are in *Exp. Output-OfAlgorithms* in [3]. The figure highlights important differences when comparing the heuristics against the four categories: (1) *spaCy* consistently assigns the highest score out of the tested heuristics; (2) the *domain model* heuristic generally assigns the lowest score; (3) for the category *Cross*, the algorithms consistently score higher than the *gold standard*; and (4) in the category *Second*, which is linked to the is-a relationship, BERT seems to correlate very well with the gold standard. Furthermore, while SpaCy gives the highest scores to the compounds that share the second word, the domain model heuristic consistently assigns the lowest scores, thereby requiring some tuning. As a side note, four of the most difficult pairs to compare (P18, P5, P12, P14) are also among the most similar pairs in the gold standard.

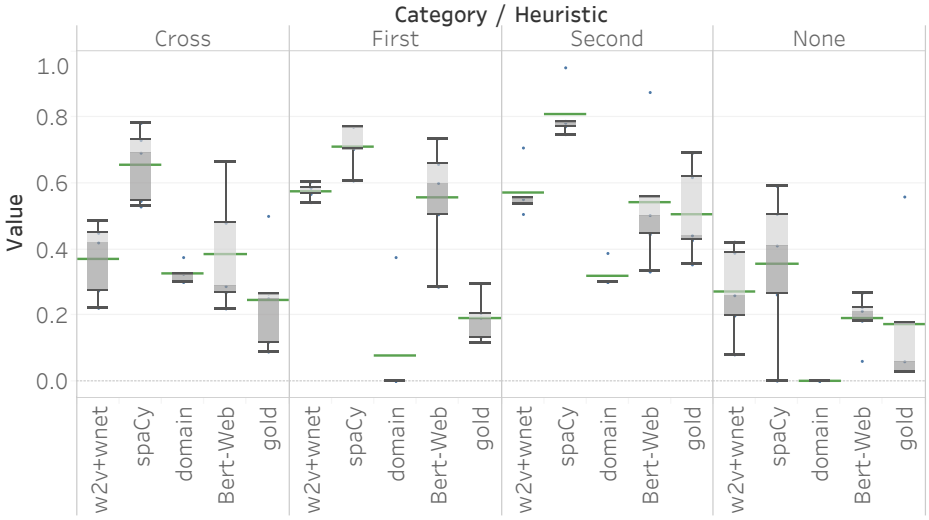


Fig. 6: Box-plot that compares the different heuristics and the gold standard, grouped by category. The green, horizontal lines denote the average value for a given algorithm and a specific category.

Standard correlation analysis. We apply the 2-tailed Spearman correlation ρ, the standard method utilized by the semantic similarity community [5]. This is adequate even in our small sample: we assessed that the gold standard is normally distributed using the Shapiro-Wilk test ($W = 0.927$, $p = 0.134$), then we visually inspected the relationship between each pair of treatments (e.g., spaCy vs. gold, domain model vs. gold) and we identified a non-linear, monotonic relationship. The results of Table 1, columns ρ and Sig. p, provide an initial answer: spaCy has a *strong* positive correlation with the gold standard (GS) ($0.6 < \rho < 0.79$, significance $p < 0.01$), both word2Vec + WordNet (w + W) and BERT-Web (B-W) have a *moderate* positive correlation ($0.4 < \rho < 0.59$, significance at $p < 0.05$), while we found *no* significant correlation between the domain

Table 1: Samples correlations (Spearman's ρ and significance p) and Euclidean distance d between the treatments and the gold standard, $n = 20$. Legend: * denotes significance at $p < 0.05$ and ** at $p < 0.01$.

Sample 1	Sample 2	ρ	Sig. p	d	Sample 1	Sample 2	ρ	Sig. p	d
w + W	GS	0.495*	0.026	1.101	w + W	B-W	0.833**	0.000	0.643
spaCy	GS	0.731**	0.000	1.804	w + W	DM	0.086	0.720	1.496
B-W	GS	0.473*	0.035	1.203	spaCy	DM	.449*	0.047	2.194
DM	GS	0.347	0.134	0.984	B-W	DM	0.256	0.276	1.472
w + W	spaCy	0.788**	0.000	0.935	B-W	spaCy	0.747**	0.000	1.196

model (DM) heuristic and the gold standard. When analyzing between-algorithms correlations, we found (i) a *strong* positive correlation between word2Vec + WordNet and spaCy ($p < 0.01$), and (ii) a *very strong* positive correlation between word2Vec + WordNet and BERT-Web. The first correlation can be explained since Alg. 1 invokes spaCy's implementation of word2Vec. The second correlation, instead, shows alignment between our algorithm and the state-of-the-art in NLP, i.e., BERT-based solutions.

Use-case specific correlation analysis. Spearman's ρ identifies correlations regardless of the actual value in the algorithms' range: for example, if an algorithm assigns consistency 0.8 to all samples, and another assigns 0.2 to all samples, the correlation will be perfect. In our use case, however, this makes a difference, for we are interested in algorithms that identify concepts to be recommended to modelers (see Sec. 2). Therefore, we apply the Euclidean distance metric as a better measure of how close the values for each couple of words are, compared to the gold standard. In this case, spaCy turns out to be the worst ($d = 1.804$): this is confirmed by visually inspecting the box-plots in Fig. 6. The algorithm that best resembles human tagging is the domain model approach ($d = 0.984$), followed by our other heuristic based on word2Vec and WordNet, and then BERT-Web. These results seem to indicate that the development of specific heuristics for compounds lead to closer results to the gold standard than off-the-shelf similarity techniques, although modern approaches like BERT offer relatively good performance.

5 Related Work

Viewpoints. The need of analyzing the perspectives of multiple stakeholders in a project is well known in requirements engineering [26]. Fischer *et al.* [13] define a *viewpoint* as a language that reflects certain aspects of a *meta-model*, while a *view* presents a model according to a viewpoint. Several tools support viewpoints and views, e.g., Sirius [33] which builds on the Eclipse stack, or MetaEdit+ [12]. Such works are the modeling infrastructure on top of which the Concept Suggester can be plugged.

Model alignment. Sànchez-Ferreres *et al.* [27] align textual and graphical representations of the same processes using NLP, machine learning, and integer linear programming. Similarly, van der Aa *et al.* [1] detect inconsistencies between the text and model of the same process. Delfmann *et al.* [10] adopts an NLP powered approach and naming

conventions. Our approach focuses on multi-model alignment and does not impose any restrictions on model labels.

Matching. Ontology matching deals with matching multiple ontologies [29]. Schema matching identifies semantically related objects in databases [15] whereas process matching aims to detect common activities or similar processes [4]. Our research problem is different, for detecting structural similarity and matching the underlying schema of models are out of our scope. We provide a lightweight solution to support collaboration by providing suggestions from a domain model.

Real-time collaboration. Some infrastructures support a near real-time, web-enabled, collaboration among modelers. Nicolaescu *et al*. [24] use a shared meta-model to generate visual model editors for a given viewpoint, they enable modelers to create views for a viewpoint and propose algorithms for managing shared editing conflicts. Debreceni *et al.* [9] focus on asynchronous collaboration, one can lock model chunks based on properties to be preserved. While these works focus on collaboration on the same model, we support modelers in the same project who do not share the same model.

Model labels. Kögel *et al.* [19] propose initial ideas on recommending modelers which elements may have to be changed when the same modeler alters a namesake or referenced element in another model. Grammel *et al.* [16] generate trace links between models by checking their similarity. They propose three similarity measures that rely on the number of namesake attributes, the number of shared parent and children nodes, and that follow the instance-of relationships. While we share similar ideas, we focus on the collaboration among multiple modelers and assume no meta-model knowledge.

Similarity. Calculating the similarity of compounds significantly less explored than the similarity of single words. One family of techniques combines the lexical similarity between the first and second words of the compounds [8,28]. Our Alg. 2 combines the lexical similarity of not only the first and second words but all combinations of word pairs of the compounds. An alternative family of techniques consider two pairs of compounds similar if they are mentioned in similar contexts [32] within a large corpus [32]; instead, Alg. 3 consults a domain model which has fewer words than a corpus.

6 Discussion and Conclusions

We have presented an approach that helps modelers in a collaborative modeling project to align their models without requiring a shared meta-model or mandating the use of identical labels. This adds flexibility to a project where multiple modeling languages are used since it removes to adjust the shared meta-model. Our approach also limits the data sharing, only the modeled concepts are shared with the service, not how concepts are connected to each other. Also, there is no direct model sharing with the other modelers. This supports competitive collaboration situations. We have implemented our approach as a web service and integrated it to the PACAS collaborative modeling platform.

We detailed two heuristics that determine the similarity of compound nouns, which are frequently used in conceptual models.

Conclusions. Our experiment shows that, when measuring the similarity of compound words, spaCy's implementation of word2Vec correlates the highest with the gold stan-

dard followed by our techniques based on word embeddings and domain models. Although preliminary, this could be a valuable finding for the information systems community, for a machine learning algorithm based on term co-occurrence in a general-purpose corpus showed better results than heuristics that make use of domain models. Our algorithms are a preliminary attempt to support the analysis and comparison of model labels that contain compound words; this research topic is under-explored.

Threats to Validity. Conclusion. Low statistical power is the major threat for our experiment: we experimented with only 5 pairs of compounds per category. *Internal.* Maturation may have occurred, as the survey respondents form their understanding of how to assess similarity as they answer the questions. To mitigate this effect, we presented the pairs in a random order. *Construct.* Mono-operation bias exists due to the choice of one specific case. Also, we intentionally decided not to give a definition of similarity, for that may introduce a bias in favor of some heuristics. *External.* While we asked the respondents to assess the similarity, the notion of similar depends on the chosen similarity metric (e.g., looking for synonyms, finding related concepts).

Future Work. We are currently designing an experiment where the subjects are given suggestions while they model. Additional research is necessary to assess if spaCy's off-the-shelf implementation outperforms our algorithms, or this is rather due to the choice of inadequate weights (e.g., σ, γ, δ in Sec. 3.1) for our algorithms.

References

1. van der Aa, H., Leopold, H., Reijers, H.A.: Comparing textual descriptions to process models - the automatic detection of inconsistencies. Information Systems **64**, 447–460 (2017)
2. Aydemir, F.B., Dalpiaz, F.: Towards Aligning Multi-Concern Models via NLP. In: Proc. MoDRE-RE (2017)
3. Aydemir, F.B., Dalpiaz, F.: Online appendix: Supporting collaborative modelling via NLP. figshare, https://figshare.com/s/e4a13da8404bcb74e0a0
4. Beheshti, S.M.R., Benatallah, B., Sakr, S., Grigori, D., Motahari-Nezhad, H.R., Barukh, M.C., Gater, A., Ryu, S.H.: Process Matching Techniques, pp. 61–90. Springer International Publishing (2016)
5. Cer, D.M., Diab, M.T., Agirre, E., Lopez-Gazpio, I., Specia, L.: Semeval-2017 task 1: Semantic textual similarity - multilingual and cross-lingual focused evaluation. CoRR **abs/1708.00055** (2017), http://arxiv.org/abs/1708.00055
6. Claasen, R.: SimCom: Measuring similarity of compound terms (2017), B.Sc. Thesis, https://github.com/RELabUU/concept-suggestor
7. Clarke, S., Baniassad, E.: Aspect-oriented analysis and design. Addison-Wesley (2005)
8. Curran, J.R.: From distributional to semantic similarity (2004)
9. Debreceni, C., Bergmann, G., Ráth, I., Varró, D.: Property-based locking in collaborative modeling. In: Proc. of MODELS. pp. 199–209 (2017)
10. Delfmann, P., Herwig, S., Lis, L.: Unified enterprise knowledge representation with conceptual models-capturing corporate language in naming conventions. ICIS p. 45 (2009)
11. Euzenat, J., Shvaiko, P., et al.: Ontology matching, vol. 18. Springer (2007)
12. Fill, H.G., Karagiannis, D.: On the conceptualisation of modelling methods using the ADOxx meta modelling platform. Enterprise Modelling and Information Systems Architectures **8**(1), 4–25 (2015)

13. Fischer, K., Panfilenko, D., Krumeich, J., Born, M., Desfray, P.: Viewpoint-based modeling-towards defining the viewpoint concept and implications for supporting modeling tools. In: Proc. of EMISA. pp. 123–136 (2012)
14. France, R., Ray, I., Georg, G., Ghosh, S.: Aspect-oriented approach to early design modelling. IEE Proceedings-Software **151**(4), 173–185 (2004)
15. Gal, A.: Uncertain schema matching. Synthesis Lectures on Data Management **3**(1), 1–97 (2011)
16. Grammel, B., Kastenholz, S., Voigt, K.: Model matching for trace link generation in model-driven software development. In: Proc. of MODELS. pp. 609–625 (2012)
17. Harispe, S., Ranwez, S., Janaqi, S., Montmain, J.: Semantic similarity from natural language and ontology analysis. Synthesis Lect. on Human Language Technologies **8**(1), 1–254 (2015)
18. Kenter, T., De Rijke, M.: Short text similarity with word embeddings. In: Proc. of CIKM. pp. 1411–1420 (2015)
19. Kögel, S., Groner, R., Tichy, M.: Automatic change recommendation of models and meta models based on change histories. In: Proc. of ME@MODELS. pp. 14–19 (2016)
20. Lapata, M.: The disambiguation of nominalizations. Computational Linguistics **28**(3), 357–388 (2002)
21. Leopold, H., van der Aa, H., Offenberg, J., Reijers, H.A.: Using hidden markov models for the accurate linguistic analysis of process model activity labels. Information Systems **83**, 30–39 (2019)
22. Levi, J.N.: The syntax and semantics of complex nominals. Academic Press (1978)
23. Lucassen, G., Robeer, M., Dalpiaz, F., van der Werf, J.M.E.M., Brinkkemper, S.: Extracting conceptual models from user stories with visual narrator. Requirements Engineering **22**(3), 339–358 (Sep 2017)
24. Nicolaescu, P., Rosenstengel, M., Derntl, M., Klamma, R., Jarke, M.: Near real-time collaborative modeling for view-based web information systems engineering. Information Systems **74**, 23–39 (2018)
25. Northrop, L., Feiler, P., Gabriel, R.P., Goodenough, J., Linger, R., Longstaff, T., Kazman, R., Klein, M., Schmidt, D., Sullivan, K., et al.: Ultra-large-scale systems: The software challenge of the future. Tech. rep., Software Engineering Institute, Carnegie Mellon University (2006)
26. Nuseibeh, B., Kramer, J., Finkelsteiin, A.: Viewpoints: Meaningful relationships are difficult! In: Proc. of ICSE. pp. 676–681 (2003)
27. Sànchez-Ferreres, J., Carmona, J., Padró, L.: Aligning textual and graphical descriptions of processes through ILP techniques. In: Proc. of CAiSE (2017)
28. Séaghdha, D.O., Copestake, A.: Using lexical and relational similarity to classify semantic relations. In: Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics. pp. 621–629 (2009)
29. Shvaiko, P., Euzenat, J.: Ontology matching: state of the art and future challenges. IEEE Transactions on knowledge and data engineering **25**(1), 158–176 (2011)
30. Sommerville, I., Cliff, D., Calinescu, R., Keen, J., Kelly, T., Kwiatkowska, M., Mcdermid, J., Paige, R.: Large-scale complex IT systems. Communications of the ACM **55**(7), 71 (2012)
31. Trask, A., Michalak, P., Liu, J.: sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. CoRR **abs/1511.06388** (2015)
32. Turney, P.D.: Similarity of semantic relations. Computational Linguistics **32**(3), 379–416 (2006)
33. Viyović, V., Maksimović, M., Perisić, B.: Sirius: A rapid development of DSM graphical editor. In: Proc. of INES. pp. 233–238 (2014)
34. Whittle, J., Jayaraman, P., Elkhodary, A., Moreira, A., Araújo, J.: MATA: A unified approach for composing UML aspect models based on graph transformation. In: Transactions on Aspect-Oriented Software Development VI, pp. 191–237 (2009)