Drilling Knowledge Bases for Hidden Frames

João GONÇALVES [1] and Pedro MARTINS and Amílcar CARDOSO
CISUC, Department of Informatics Engineering, University of Coimbra

Abstract. Sometimes we may wonder where we can obtain semantic frames to be used in a computational model of Conceptual Blending. The frames can be hand-made, adapted from existing frame libraries or, as we suggest in this document, discovered by mining conceptual graphs. The idea we summarise here depicts our proposed computational mechanism to explore the knowledge base containing the conceptual graphs for recurring semantic patterns representing potential frames.

Keywords. frames, conceptual graphs, data mining, conceptual blending

1. Introduction

A general issue in Computational Creativity (CC) arises from the need of having data crucial to build models. With the ambition of going beyond simple toy problems we are currently looking forward for the CC prototype system we have been developing to be capable of handling larger amounts of data. That system is comprised of various modules: a fast mapping module [1] and a blending module [2] (the Blender) based on Conceptual Blending (CB) theory [3].

The mapping module is implemented as a Genetic Algorithm (GA) that extracts mappings from large-scale conceptual graphs such as ConceptNet [4] or from the NELL [5]. These conceptual graphs are represented as semantic graphs where vertices are concepts and the edges correspond to relations between concepts, that is, a graph structure representing Subject-Verb-Object (or equivalent) triples. The same data functions as representations of the input spaces in the blending module. This module also requires a set of one-to-one mappings between concepts (representing analogies) and frames to guide the blend towards having specific aspects [6]. The source code for all the above modules (and new ones to be researched in the future) is available at `https://github.com/jcfgonc/phd`.

2. Conceptual Blending

Fauconnier and Turner [3] proposed CB as a cognitive theory to explain cognitive-linguistic phenomena such as analogy, metaphor, metonymy or counterfactual reasoning [7]. In CB, the generation of new ideas is done through the integration of existing ones.

[1] Corresponding author: jcgonc,pjmm,amilcar}@dei.uc.pt

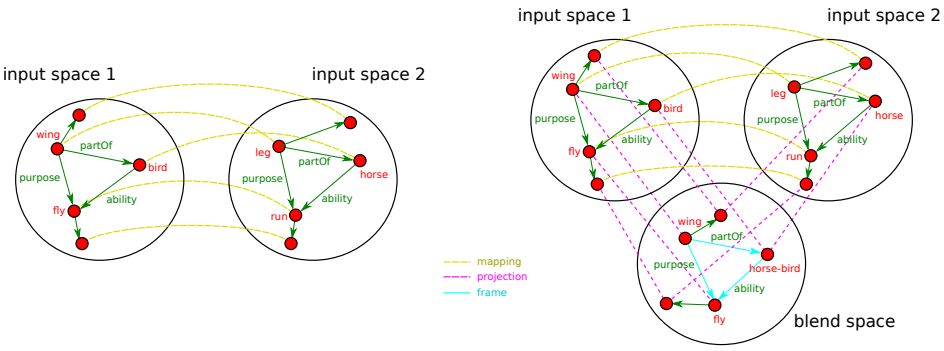

Figure 1. The idea behind Conceptual Blending (CB). In the left an example of a mapping of concepts between two domains, based on the same structure of edges (relations) connecting the concepts - a graph isomorphism. In the right an example of a CB process using the same mapping to create a new blend space containing the frame shown in cyan. Best seen in colour.

Various forces act to connect and blend different parts of knowledge present in two or more input spaces, including a generic space containing elements common to both the input spaces and representing a shared conceptual structure between them [8]. These spaces serve the purpose of being initial sources of information. Many works in CC are focused on computational models which explore CB as a way of creating new knowledge [9, 10].

A CB framework as the one shown in Fig. 1 requires at least two input mental spaces containing the source of information to be processed. These input spaces could correspond to semantic networks comprised of relations between concepts, i.e. Knowledge Bases (KBs). One of the steps CB does - named composition - is using an alignment of concepts of each input mental space as a guide of which concepts are to be used in the blend space. In the end, a new mental space emerges from the CB process containing its "output".

For the guiding alignment stated above, some CB implementations, including our Blender, use a graph isomorphism between the input spaces to align concepts with a similar structure (Left Fig. 1). Thanks to that alignment, part (or all) of the structure present in the input spaces is also present in the blend space (shown in the right Fig. 1). The structures describing the alignment of concepts are termed mappings and are comprised of sets of concepts. Many tools have been developed for that purpose, such as the Structure Mapping Engine [11], Sapper [12] and a fast stochastic algorithm [1] capable of handling semantic networks in the order of millions of relations.

In addition to the composition step, additional steps are executed to complete and elaborate the blend, namely, completion and elaboration [3]. The latter involves executing additional logic (such as rules) present in the input spaces or in the generic space, hence elaborating (inferring) into the blend space new knowledge. In the completion step the structures stored in the blend space are completed with background knowledge such as frames to generate consistent and meaningful structures [8, 7].

It is in this step that we observe the importance of frames - to give the blend a recognisable meaning such as an event or an entity, e.g., the journey of someone in the Buddhist Monk example [13]. In Fig. 1 the blend space contains a frame drawn in cyan. That frame has specific meaning - giving an entity (*horse-bird*) an ability (*flying*) because the entity has a part with that purpose (*wing*). Although the input spaces also contain the

given frame, through composition the relations can be brought from the input spaces and assembled into the frame structure that emerges in the blend space.

3. Motivation

In this work we discuss our approach for solving some of the issues reported on [2], mainly regarding obtaining the semantic frames. These structures represent a pattern of relations between entities (concepts) and are rooted on previous lived experiences [14, 3]. They are required by CB to guide the blending process in order to exhibit recognisable wholes, possibly multiple ones [7]. Frames can be seen as shaping the emerging blend with one or more mental images and given the probabilistic nature of the blending process, it is highly likely that the resulting blend will contain multiple frames.

We have been using a large subset of ConceptNet V5 processed by us and representing the conceptual spaces described above. We added various new facts and removed what we found as irrelevant or controversial information including biased relations as *purposeOf(woman,cook)*. Some of these were also noted by [15], including incomplete or erroneous relations such as *isa(prion,prokaryote)*. Our latest processed version of ConceptNet has 1229508 concepts, 1791604 relations and an improved version will be made available publicly in the future. Although there are KBs with frames such as FrameNet [16], MetaNet [17] and Framester [18], we speculated whether ConceptNet or any other KB could be mined for hidden patterns representing possible frames. For the moment we have focused on the mining topic and that is the purpose of this paper.

4. Approach

The idea makes use of a GA to find repeating structures in the KB resembling frames. Currently, these structures ignore specific concepts (such as bird or plant) and are only based on the type of relation. They can be seen as depicting geometric structures in the semantic graph. Based on our previous experience with handling large-scale semantic graphs and the complexity of this problem, we think that the best strategy to find acceptable solutions in a useful time is to use stochastic algorithms such as GAs. Additionally, our GA is mostly concurrent, makes use of multi-core processors and is able to search the KB for patterns almost in real-time.

Frames are stated as Prolog queries and are applied to the relations stored in the KB. An example of a frame used by the CB module is the idea of some entity A containing a part B whose purpose is to C, hence A gains the C ability. If variable $C =$ 'fly' this frame could represent the idea of an entity with a part granting it the ability to fly. This frame can be stated as the rule:

ability(A,C), purpose(B,C), partOf(B,A).

This is equivalent to a three edge, three concept semantic graph in the form of a closed triangle. With a querying engine (i.e., Prolog or Datalog based), the KB can be searched for solutions for the above frame and thus if it is a prevalent pattern or not. Solutions are counted as different instantiations of variables. An example of the depicted pattern and of a small semantic graph containing six occurrences of the same pattern is shown in Fig. 2.

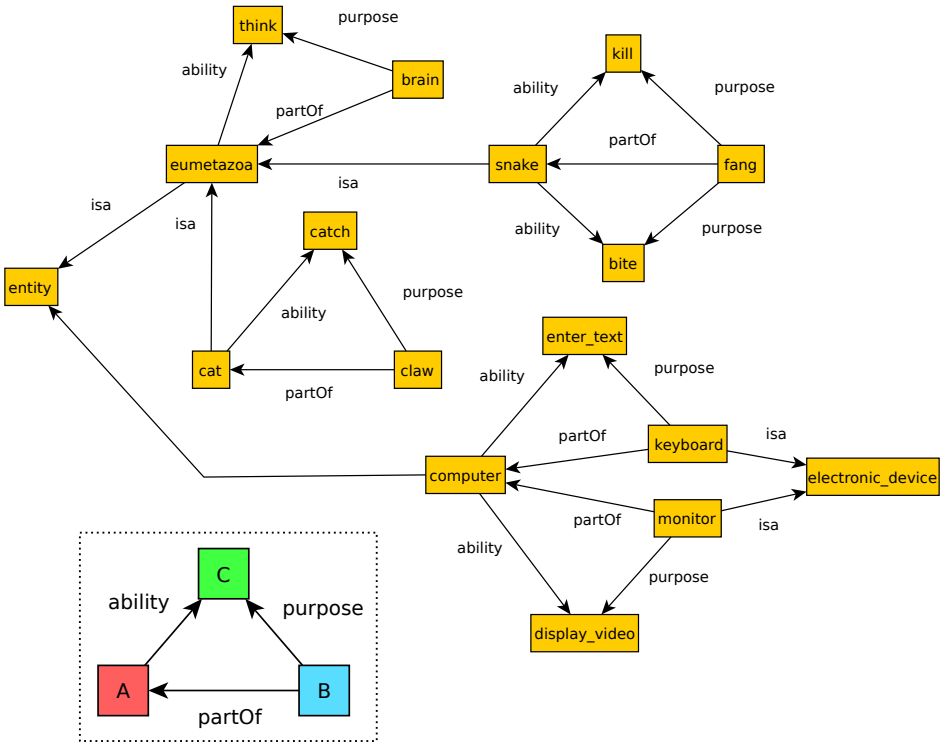

Figure 2. An example of a three relation pattern (bottom-left) as well as a small conceptual space containing six occurrences of the same pattern (top). Best seen in colour.

4.1. Genetic Algorithm details

As with all GAs, ours evolves a population of chromosomes during multiple epochs. The idea is to examine, traverse and scale different regions of the KB by distributing this process amongst multiple individuals. In each epoch, a chromosome is mutated, has its fitness evaluated and possibly selected for inclusion in the next epoch. The selection procedure is a simple binary tournament between two members of the population of the same epoch.

A potential pattern (i.e. frame) is stored in an individual chromosome of the population as a directed semantic graph. This graph is mapped to the corresponding query by replacing each concept present in the chromosome's semantic graph by a unique variable (e.g. *bird* $\rightarrow$ *B*). It is important to note that the graph stored in the chromosome contains concepts and not variables, as concepts are required to match the chromosome's graph to a region of the KB when applying the mutation operator.

In the first epoch, the chromosomes are small regions extracted from the KB with two-three connected relations (on average). The reason for this is that, in our opinion, frames with one relation do not seem to be useful for the blending process of the blender module. For a similar reason, there is no need to use such a tool as this one to check for the most present relation in the KB if single relation frames are allowed en masse.

The stochastic search of the GA is implemented with a mutation operator, which adds or removes relations to the semantic graph stored in the chromosome. New relations

are copied from the KB according to common concepts between the chromosome's graph and the KB. Hence, an addition chooses a random concept in the pattern, checks its relations in the KB and adds one of these at random to the pattern. Relation removal is simpler, they are randomly removed from the pattern with no specific criteria and in the case the pattern gets fragmented, only a single random component of the pattern's graph is maintained.

Given the stochastic nature of the mutation operator, each evolving pattern may degenerate into multiple graph components, that is, it may transform into various disconnected sets of relations. The mutation is always followed by a repairing operation which guarantees that the pattern is composed of at most one graph component. The repairing algorithm is simple, it remove all except one of the graph components. This remaining component is randomly chosen in order to maximise the randomness of the GA.

We use a fast querying tool named *querykb* provided by Aaron Bembenek which at this moment is temporarily unavailable. It was developed to strictly count all possible solutions to a given query and does not instantiate variables. However, for its purpose of counting solutions, it is exceptionally fast and efficient. It is invoked whenever the GA evaluates a chromosome's fitness function. The number of pattern matches can reach extremely high numbers and therefore are returned by the tool as Java BigIntegers.

4.2. Fitness function

The score of each evolved pattern is calculated with the fitness function, applied to every chromosome in the current GA population and thus, every pattern has an individual score. As the GA is (currently) implemented as a single objective optimisation task, we had to combine k multiple objectives in a single fitness function f using a weighted sum of individual components f_i (a linear aggregation). The weights are scalars and are manually chosen to fine-tune the GA and its results towards showing aspects we want the patterns to exhibit. At the moment, we have the following four objectives ($k = 4$):

1. f_1 is the logarithm to the base 10 of the number of matches of the pattern in the KB;
2. f_2 is the number of relations (edges) the pattern has;
3. f_3 is the number of different (unique) relation labels the pattern has;
4. f_4 is the standard deviation of the histogram of relations the pattern has.

The first objective, f_1, drives the optimisation towards finding the most recurring pattern within the KB, without any regard to the pattern's structure. Because of the colossal amount of matches for many patterns, to include this objective in the fitness function (stored as a double floating number) the code returns the logarithm to the base 10 of the total matches (a BigInteger number) returned by the querykb tool. We decided on base 10 to be easier to deduce from the logarithm the number of digits the pattern match count has.

In addition, we include structural information regarding the pattern as three more objectives. Within those, objectives f_2 and f_3 guide the optimisation towards finding larger patterns. Objective f_2 has more impact on that purpose compared to f_3 but does not guarantee that the pattern has a diverse amount of relations, an aspect controlled by the latter objective.

Objective f_4 reflects the count of relations of each type (label) contained in the pattern, i.e., four relations with the label *partof*, two relations with the *causes* label, etc.

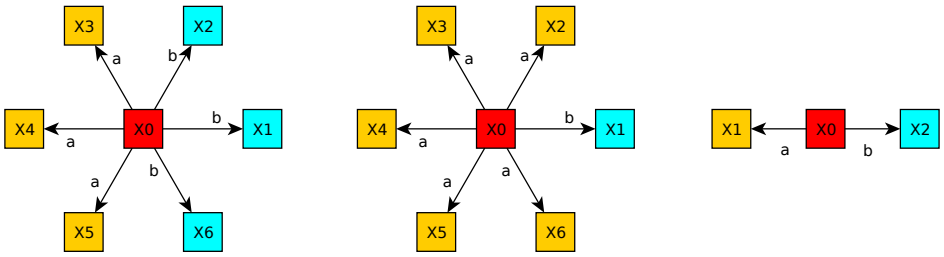

Figure 3. Example of pattern generalisation. The first two have six relations of two types: *a* and *b*. The third pattern has the minimum amount (2) of relations with the same two types. Best seen in colour.

Relations not existing in the pattern are not included in the histogram calculation. The purpose of this objective is to force the pattern to have a balanced amount of relations of different labels (Fig. 3). Adjusting this weight in the opposite sense will compel the pattern to have a majority of relations of the same label. Hence, this objective controls the generalisation of patterns: generic patterns are easily or more commonly instantiated. Using Fig. 3 as a reference, structures matched by the pattern on the left are also matched by the pattern on the right. The opposite is not true, as the left pattern requires six distinct relations and seven distinct concepts when compared to two relations and three concepts in the right pattern. Therefore, the leftmost pattern is a specialisation of the rightmost pattern. Although objective f_4 helps in this regard, it will only drive the GA towards generic frames given a constant f_3, that is, they must have the same relation types (i.e. two in Fig. 3).

5. Current Results

We removed from ConceptNet V5 four relations that in our opinion will not be very fruitful in the Conceptual Blending process: *isa*, *derivedfrom*, *synonym* and *similarto*. These relations are very generic and the remaining 35 relations contain more specific information regarding the concepts they are related to.

Problematic concepts with non-ASCII characters were also removed because of incompatibilities with some Java libraries and the handling of Unicode characters. Consequently, the complexity of the mining problem was reduced and the working KB had 377719 relations and 278921 concepts.

The querykb tool allows the user to set a time limit to the query/pattern match counting process, something we had to set at 30 minutes per pattern to have results within an acceptable time limit. The tool by itself is concurrent and forces our GA to be sequential when calculating the fitness function for all the chromosomes. This sets a limit that in the worst case only one chromosome would be evaluated per 30 minutes and hence, a maximum number of epochs to be processed by the GA in a given time. Unless otherwise specified, the weights used in balancing the four objectives were the following:

weights	w_1	w_2	w_3	w_4
values	0.1	0.1	1	1

We ran 10 experiments with an execution time of 48 ± 24 hours and 70 ± 30 epochs per experiment. The population size (number of chromosomes/patterns) was constant

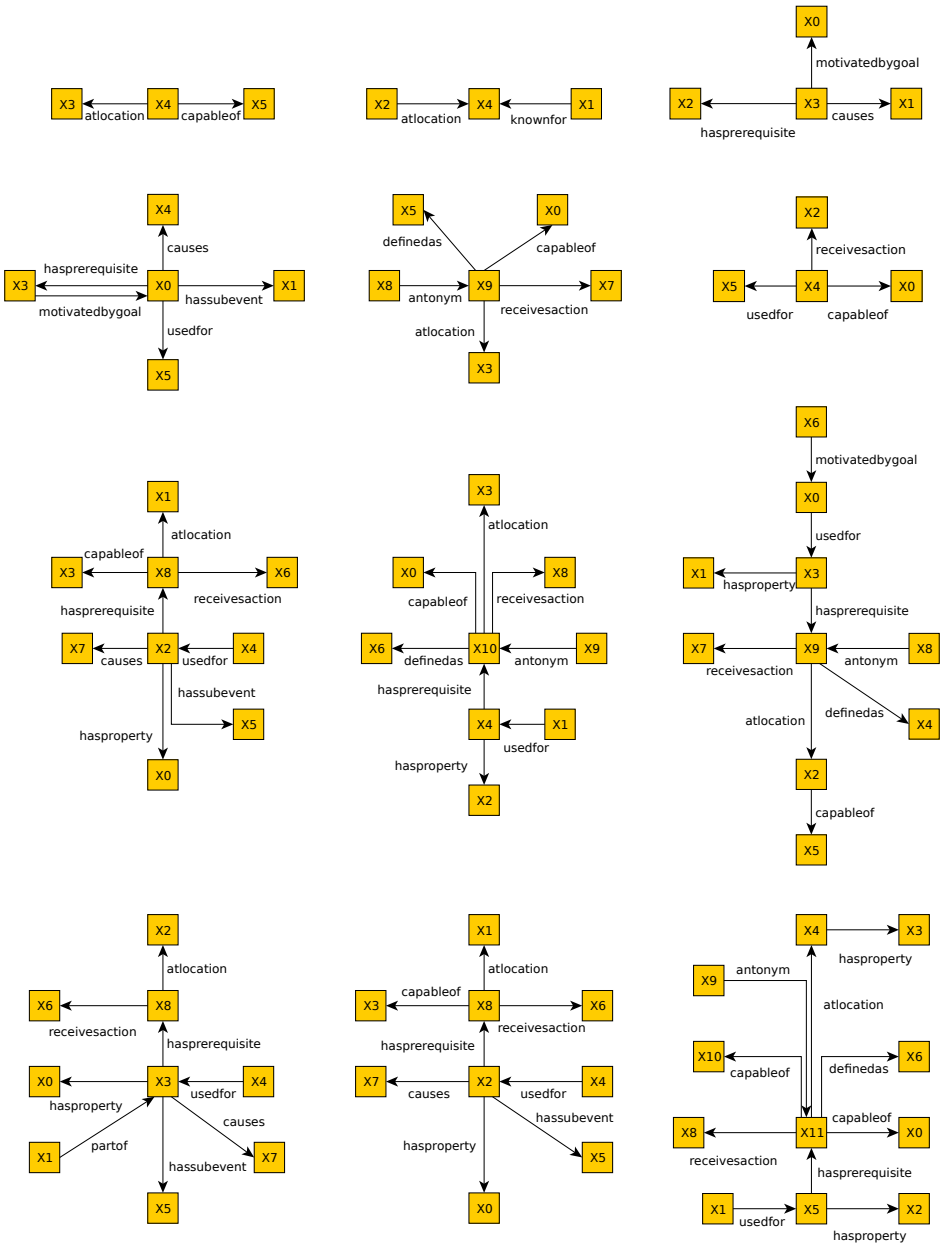

Figure 4. Examples of patterns we found most interesting during the various executions of the algorithm. They are ordered top to bottom according to their number of relations.

per epoch at 64 elements. The GA used a binary selection tournament with the winning probability of the strongest candidate being 75%. The machine used in the experiments had two eight-core Xeon E5-2667 v2 processors (total 16 cores) working at 3.6 GHz, 64 GB of RAM and the Microsoft Windows 7 SP1 operating system. The querykb tool used a block size of 256 and 32 threads.

Fig. 4 contains twelve of the patterns we found interesting. Changing the weight of objective f_2 had an influence on the number of relations in the patterns, which can be seen by watching the top to bottom ordering of the patterns in Fig. 4. Objective f_4 had the effect of lowering the presence of multiple relations of the same type, although it does not guarantee the lowest amount as seen in the last pattern containing two *hasproperty* relations.

We were not able to extract graph structures with cycles similar to the triangle pattern shown in Fig. 2. The search algorithm does not have any reason to find those types of structures and therefore they do not emerge during the GA's execution. It is our opinion that this may be solved with some changes in the mutation and with one or more additional objective function(s).

As we are currently lacking semantic quality measures to evaluate the patterns, except for the four objectives stated before we can not firmly conclude if any one of the patterns is better or worse than the others either in the CB process either for any other purpose.

6. Conclusion and Future Work

We presented a mechanism to mine for repeating patterns in a KB. The purpose is to find potential frames in semantic graphs that are used by Conceptual Blending modules. At the moment we have a working prototype that finds recurring patterns representing prototypes of frames. The prototype is ready to be evaluated semantically and to see whether it matches the creative requirements expected to arise ahead in our future work. We will also study other objectives to guide the quality of the emerging patterns according to whatever requirements we think the frames should have. This will require further understanding of frames and their impact on the resulting blends of the CB module.

Acknowledgements. João Gonçalves is funded by Fundação para a Ciência e Tecnologia (FCT), Portugal, under the PhD grant SFRH/BD/133107/2017. We greatly acknowledge Aaron Bembenek for his impressive querykb tool and his availability for matching the tool to the needs of our project. To a truly generous person, a sincere thanks.

References

[1] João Gonçalves, Pedro Martins, and Amílcar Cardoso. A fast mapper as a foundation for forthcoming conceptual blending experiments. *Special Track Analogy - Proceedings from The Twenty-Sixth International Conference on Case-Based Reasoning*, 2018.

[2] João Gonçalves, Pedro Martins, and Amílcar Cardoso. Blend city, blendville. In *Proceedings of the Eighth International Conference on Computational Creativity*, 2017.

[3] Gilles Fauconnier and Mark Turner. *The Way We Think*. New York: Basic Books, 2002.

[4] Robert Speer and Catherine Havasi. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686, 2012.

[5] T. Mitchell et al. Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*, 2015.

[6] Graeme Ritchie. Some empirical criteria for attributing creativity to a computer program. *Minds and Machines*, 17(1):67–99, 2007.

[7] Francisco Câmara Pereira. *Creativity and artificial intelligence: a conceptual blending approach*. Berlin: Mouton de Gruyter, 2007.

[8] Pedro Martins, Pereira Francisco, and Amílcar Cardoso. *The Nuts and Bolts of of Conceptual Blending: Multi-Domain Concept Creation with Divago*. Springer, 2017.

[9] Martin Žnidaršič et al. Computational creativity infrastructure for online software composition: A conceptual blending use case. In *Proceedings of the Seventh International Conference on Computational Creativity (ICCC 2016)*, Paris, France, 2016. Sony CSL, Sony CSL.

[10] Marco Schorlemmer et al. Coinvent: Towards a computational concept invention theory. In *Proceedings of the 5th Int. Conference on Computational Creativity, ICCC-14, Ljubljana, Slovenia*, 2014.

[11] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[12] Tony Veale and Mark Keane. The competence of sub-optimal structure mapping on hard analogies. In *Proceedings of the International Joint Conference on Artificial Intelligence*. IJCAI-97, 1997.

[13] Gilles Fauconnier and Mark Turner. Conceptual integration networks. *Cognitive Science*, 22(2):133–187, 1998.

[14] Charles J. Fillmore. Frames and the semantics of understanding. *Quaderni di Semantica*, 6(2):222–254, 1985.

[15] Atilim Günes Baydin, Ramon López de Mántaras, and Santiago Ontañón. Automated generation of cross-domain analogies via evolutionary computation. *CoRR*, 2012.

[16] Josef Ruppenhofer, Michael Ellsworth, Miriam Petruck, Christopher R Johnson, and Jan Scheffczyk. *FrameNet II: Extended theory and practice*. 2016.

[17] Oana David, Ellen Dodge, J Hong, ELISE Stickles, and E Sweetser. Building the metanet metaphor repository: The natural symbiosis of metaphor analysis and construction grammar. In *The 8th International Conference on Construction Grammar (ICCG 8), Osnabrück, Germany*, 2014.

[18] Aldo Gangemi, Mehwish Alam, Luigi Asprino, Valentina Presutti, and Diego Reforgiato Recupero. Framester: a wide coverage linguistic linked data hub. In *European Knowledge Acquisition Workshop*, pages 239–254. Springer, 2016.