Automating the Encoding for the LISA Model of Analogy from Raw Text

SEAN WILNER, University of Illinois Urbana-Champaign
JE HUMMEL, University of Illinois Urbana-Champaign

Analogy is an integral part of human cognition. Consequently researchers would like to produce computational models of analogy to test implementations of theoretical claims. Such computational models have relied almost exclusively upon hand-coded representations, making the resulting models too dependent upon the modeler's choices, and thus difficult to interpret in isolation of the modeler. In an effort to combat this dependency, we present here a means of automatic encoding for the LISA model of analogy and inference.

General Terms: Analogy, LISA, NLP

Additional Key Words and Phrases: Natural Language Processing, Computational Linguistics, Cognitive Psychology, Analogy

1 INTRODUCTION

Analogy, and more generally, relational reasoning are core cognitive capacities [6, 8, 10] that, along with language, seem to separate human cognition from the cognitive abilities of all the other great apes [22]. Many of the computational models of analogy arising from machine learning revolve around solving semantic, or 4-way, analogies of the form "king:queen::brother:?" where the model fills in "sister" [19]. These models fail to account for the more substantial form of analogy – narrative analogy – wherein agents and actions must all be analogically mapped to one another. Numerous models have been developed to account for our ability to make such analogies. Among the most influential have been Falkenhainer et al.'s [3] Structure Mapping Engine (SME), Holyoak and Thagard's [9] Analogical Constraint Mapping Engine (ACME) and Hummel and Holyoak's [11, 12] Learning and Inference with Schemas and Analogies (LISA) (see Gentner & Forbus, 2011[7], for a review).

Although these and other models account for many aspects of human relational thought, they share a common limitation: All of them rely on hand-coded knowledge representations, placing a great deal of control in the modeler's hands. As such, it is often unclear how many of the successes of these models reflect the models' fundamental tenets and how many reflect arbitrary representational and modeling choices made by the modeler at the time of creating the simulations (see, e.g., Lu et al. 2008 [17]). Since such models are only as good as their modeler, the support or challenge to any theoretical claims they make must be interpreted carefully.

Ideally, these models would produce their representational inputs directly from natural language, removing their dependence upon hand-coded inputs. There are some models that approach this goal for specific narrow domains, such as intuitive physics (Friedman SE & Forbus KD, 2009 [5]) and moral decision making (Dehghani M et al., 2008 [2]). However, the task of generalizing open domain natural language for analogy remains elusive. To address this problem, we describe recent efforts applying several Natural Language Processing (NLP) techniques

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA9550-12-1-0003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.
Author's addresses: S. Wilner, Illinois Informatics Institute, University of Illinois UrbanaChampaign; JE Hummel, Department of Psychology, University of Illinois UrbanaChaimpaign.

to the automatic generation of knowledge representations for LISA. When combined, these techniques allow us to transform English sentences into semantically-rich propositional representations usable to the LISA model of analogy (i.e., representations in "LISAese").

1.1 Representation

LISAese is a collection of semantic encodings and structural representations which, together, allow LISA to produce semantically informed, structurally consistent analogies. LISA employs semantic representations for both objects and predicates and encodes structure in a predicated logical form. Semantics for an object in LISAese consist of a distributed list of abstract "semantic units" and corresponding weights. For instance, if we were to hand-code the representation for a taxi driver, "Bob" and a bank teller "Jill", we may end up with the following semantics and corresponding weights:

Bob: human=1.0, adult=0.7, male=1.0, employed=0.8, taxi_driver=1.0

Jill: human=1.0, adult=0.7, female=1.0, employed=0.8, bank_teller=1.0

Bob's semantics could then overlap with the semantics for our bank teller, "Jill", who is also human, adult, and employed, but would differ on male and taxi driver. LISA treats these "semantic units" as universal across all representations, that is, if two "semantic units" have the same name, they are the same unit. Aside from this universality, the names are arbitrary and the choice to give them relevant names with regard to the objects they are attached to only serves to facilitate human readability.

Similarly, predicate semantics consist of "semantic units" and weights; however, predicate semantics are defined at the level of their relational roles rather than the predicate itself. For example, the two-place predicate *love* is defined by two lists of semantics. The first list represents the agent role of *love* (the lover); the second represents the patient role of *love* (the beloved). This could be represented as follows:

love:
[lover=1.0, emotion=1.0, strong_emotion=0.8, passionate=0.9]
[beloved=1.0, receives_emotion=1.0, receives_strong_emotion=0.8, receives_passion=0.9]

With defined semantics on objects and predicates, LISA is able to ground its structural representation, given in a predicated logical format, to represent "Bob loves Jill": *love*(Bob, Jill). This logical representation is robust to recursive references such as "Jill knows that Bob loves her." which yields: *know*-(Jill, *love*(Bob, Jill)). Indeed, this is sufficient to represent the necessary semantic and structural information LISA needs to make analogies and inferences.

1.2 Task

Our task in this paper is to outline a technique to produce semantic and structural representations automatically from text. Subsequently, we will show that LISA can use the resulting LISAese to successfully find analogies and inferences. More specifically, we will present a method which achieves the following:

(1) Generates Semantics
- Object semantics that account for modification by adjectives, noun-phrases, and copular verb attachments
- Predicate semantics with role-by-role representation

(2) Generate Structure

- Produces logic form
- Handles recursive references
- Associates multiple references to a single entity

By using NLP techniques to automatically pre-process and encode text into LISAese, the LISA model will no longer need to rely on hand-coded knowledge representations. Freeing it of this requirement will clarify how LISA's successes as a model reflect its fundamental tenets rather than choices made by the modeler.

2 GENERATE SEMANTICS

Ultimately, we need to generate semantic representations for both objects and predicates, but the requirements for automated generation of the two differ slightly. This is because, as previously noted, in LISAese semantics are assigned not to the predicates themselves, but rather to each predicate role independently.

2.1 Object Semantics

Objects may be represented in a fairly straightforward manner using word embedding. Word embedding is a means of mapping words into a semantic space (a vector space) such that words which are 'similar' are closer in the resulting space than words which are 'dissimilar'. There are many means of producing word embeddings. One of the most commonly used semantic embedding tools is Google's Word2Vec [19] which produces vectors based off carefully crafted word co-occurrence statistics. Using these vectors, we can produce embeddings for individual words such as "cat", "January", or "happy". However, this means of embedding can break down upon encountering proper nouns such as "Microsoft" or "Bob". In order to properly encode proper nouns, we employ an NLP technique called Named Entity Recognition (NER) which identifies what kind of entity the proper nouns are: "Microsoft"→ORGANIZATION and "Bob"→PERSON. Using these NER tags, we can encode the proper nouns by the semantic embedding of their associated tag (for added specificity, where known, we also incorporate the gender of each proper noun).

In order to accommodate various modifiers, we additively combine the semantic vectors corresponding to each object. These vectors consist of the embeddings of the words contained within or associated with an entity, for example, the words in a noun-phrase, an NER label, adjective modifiers, or copular modifiers. Copular verbs are stative verbs such as is/are/was/were, seem, or appear. For example, "smelly Sam" and "Sam is smelly" both result in the incorporation of the semantics for "smelly" in the representation of Sam.

Using the resulting vectors, we can produce LISAese by creating an arbitrary semantic label, call it "SemUnit_i", to every i^{th} index of the vector where the weight on the generated semantic unit is the value at the i^{th} index of the vector. For example, if our word embedding was 4 dimensions (instead of 300), we could have the following embedding for "happy":

happy→ [0.14, 0.0024, 0.81, -0.37]

Which would then yield a LISAese representation of "happy":

happy→ SemUnit_0=0.14, SemUnit_1=0.0024, SemUnit_2=0.81, SemUnit_3=-0.37

Thus allowing us to produce object semantics in LISAese from the Word2Vec word-embeddings.

2.2 Predicate Semantics

Predicate semantics in their most general form are more difficult to automatically encode than objects. Direct role-by-role representation via word-embedding is certainly non-obvious, although we will explore methods to do so in the future work section. Here, we will simplify our semantic representation to the easiest usable form wherein we assign semantics to each role corresponding to word embeddings for the predicate. For example, in our 4-dimensional embedding from before, we can imagine the following vector for "love":

"love" → [0.23, 0.01, 0.76, -0.45]

Copying the embedding onto the agent and patient roles of *love* gives us:

love:
[SemUnit_0agent=0.23, SemUnit_1agent=0.01, SemUnit_2agent=0.76, SemUnit_3agent=-0.45]
[SemUnit_0patient=0.23, SemUnit_1patient=0.01, SemUnit_2patient=0.76, SemUnit_3patient=-0.45]

While this representation meets all of our requirements, it does have a glaring weakness. There will never be any semantic overlap between two different predicate roles (e.g. agent and patient). The resulting LISAese will make LISA preferentially map agents and patients onto other agents and patients respectively. Again, we will explore alternatives to this encoding which do not suffer from the same weakness in the future works section below.

Using the above semantic representations, we meet all the requirements for LISAese's semantic encoding and are ready to explore how to structurally represent text in LISAese.

3 GENERATE STRUCTURE

In order to encode language in the kind of predicated logic LISAese requires, we need to identify predicate roles and the objects which fill them. In the simplest case, we may be presented with text such as "Bob loves Jill". In which there is one predicate with two roles, agent and patient, filled by "Bob" and "Jill" respectively. As mentioned earlier, this structure becomes more difficult when predicates fill the roles of other predicates as in "Jill knows Bob loves her". In this later case, the structure we hope to capture is that "Jill" is the agent and *love* is the patient of *know*. To discover these attachments, we may employ an NLP technique called Semantic Role Labeling (SRL). SRL identifies predicates and their roles as well as portions of the text which fill those roles. Using SRL, we can label "Jill" as the agent and "Bob loves her" as the patient of *know*. Similarly, "Bob" is identified as the agent, and "her" the patient of *love*. For our work, we use the SRL engine laid out in Punyakanok, Roth, and Yih [24] which uses joint inference over several SRL systems and Integer Linear Programming for global constraint satisfaction to optimize its labeling. After using our SRL engine, we have the following structural encoding: *know*(Jill, *love*(Bob, her)). This is nearly enough to properly structure the input for LISA since we are able to encode the predicates in a LISA's logical form and are able to handle recursive references to predicates as arguments to other predicates. The last remaining piece is to properly associate "Jill" with "her". In order to achieve this, we use another NLP technique called 'co-reference resolution' [18]. Co-reference resolution associates multiple references to the same entity throughout a text. Using this technique, we are able to associate "her" with "Jill", yielding our final representation: *know*(Jill, *love*(Bob, Jill)).

The automated structural encoding, along with the semantic encoding, is sufficient to produce functional LISAese without the involvement of any hand-coding and little modeler input. However, this automatic encoding is useless if LISA is not able to function on the resulting LISAese. In order to test this, we explore three benchmark simulations LISA has successfully completed in the past: Latin to Logic, Love Triangle, and Driving a Vehicle.

4 SIMULATIONS

To test our automated representations, we will compare LISA's discovered mappings and inferences on them against the hand-coded LISAese. Since our automatic encoding produces several hundred semantics for each object and role, we have omitted them from this paper. However, since there were some differences in some of the structural representations between the hand-coded and automatic representations, we have provided both for each benchmark below.

For "Latin to Logic"[14], the benchmark representation is shown below:

Analog 1:
is_incomplete(Second_Order_Logic)

Analog 2:
is_self_referential(Natural_Language)
has_unprovable_statement(Natural_Language)
is_incomplete(Natural_Language)

LISA can use this produced representation to find the analogy between Second Order Logic and Natural Language and infer that Second Order Logic has unprovable statements and that it is incomplete.

In order to test our automatic representation, we need to identify English sentences which convey a similar meaning. Running LISA over the representations generated by these sentences, we can check if it produces the correct mappings and inferences. For that purpose, we chose the following stories: "Second Order Logic is incomplete." and "Latin is able to self reference. Latin proves and disproves a statement. Latin is incomplete." Using the automatic representational system described here, we produced the following representation:

Analog 1:
be(Second_Order_Logic, incomplete)

Analog 2:
be(Latin, able_to_self_reference)
prove(Latin, a_statement)
disprove(Latin, a_statement)
be(Latin, incomplete)

Running LISA on this representation does in fact produce a mapping between Second Order Logic and Latin. LISA makes both of the desired inferences, namely that Second Order Logic is self referential and that it can prove and disprove a statement. Worth noting is that the structure generated by our automated system is quite different from that of the original representation. However, the correct mapping and inference was still produced.

For "Love Triangle" [12] the hand-coded benchmark structure and the automatic structural encoding were identical, and it is shown below:

Analog 1:
love(Sally, Jack)

love(Jack, Megan)
hate(Sally, Megan)

Analog 2:
love(John, Mary)
love(Mary, Bob)

In order to produce this representation automatically, Analog 1 was generated from the sentences "Sally loves Jack. Jack loves Megan. Sally hates Megan." and Analog 2 from "John loves Mary. Mary loves Bob".

"Latin to Logic" is the easiest of our three benchmarks since the structure is simple and semantic similarity alone is enough inform mapping. In "Love Triangle", while the structure remains simplistic, the object semantics are inversely correlated with the ideal mapping. That ideal mapping is from "Sally"→"John", "Jack"→"Mary", and "Megan"→"Bob", which is counter-indicated by semantic similarity on the subset of semantics corresponding to gender. In the hand-coded version, LISA finds this ideal mapping and produces the correct inference, namely *hates*(John, Bob) in Analog 2. After running LISA on the automatically produced LISAese, the same mapping and inference was found.

Lastly, we will look at the most complicated benchmark, "Driving a Vehicle" [12]. Its hand coded structure is shown below:

Analog 1:
has(Joan, car)
want(Joan, goto(Joan, airport))

Analog 2:
has(Sam, car)
want(Sam, goto(Sam, beach))
drive(Sam, car, beach)

The automatic encoding produced a very similar structure:
Analog 1:
have(Joan, a_car)
want(Joan, go(Joan, the_airport))

Analog 2:
have(Sam, a_jeep)
want(Sam, go(Sam, to_the_beach))
drive(Sam, a_jeep, to_the_beach)

To produce this, the NLP pre-processor was given "Joan has a car. She wants to go to the airport." for Analog 1 and "Sam has a jeep. He wants to go the beach. He drives the jeep to the beach." for Analog 2.

On both the hand-coded and auto-coded LISAese, LISA found the correct mapping that "Joan" is like "Sam" and the "jeep" is like the "car". LISA also managed to produce the correct inference that Joan drives her car to the airport.

5 CONCLUSION

Using NLP techniques to pre-process and encode text into LISAese has proven to be a fruitful approach to incorporating the encoding process into LISA as a model. With the incorporation of this linguistic front-end, LISA is now the first computational model of analogy capable of encoding simple open-domain text and performing analogical mapping and inference over it. More importantly, the LISA model no longer needs to rely on hand-coded knowledge representations. With this, it is now clearer how LISA's successes as a model reflect its fundamental tenets rather than the representational and modeling choices made by the modeler.

6 FUTURE WORK

We hope to improve both the semantic and structural encoding in the future with better semantic libraries and relational capture. For object semantics, we can use sparse over-complete vectors [4] for the word embeddings, since it has previously been shown that LISA performs better over sparse encodings. For predicate semantics, we can explore role specific semantics so that predicate role semantics can overlap between labeled roles (some agents overlap with some patients). Towards that end, we intend to explore several different tracks for embedding. We will look at PropBank [15] labels for verb specific roles and use the embeddings of the words in their labels (agent of *love* is "lover", patient of *love* is "loved"). We will also look at embedding the roles directly, either by forming role co-occurence chains and using Word2Vec on those chains directly, or by replacing the spans of text by their SRL label and running Word2Vec over the replaced text. However, regardless of what embedding techniques we use, the representation technique as we've described it falls pray to a common word embedding weakness – inability to differentiate phrasal cconstructs. That is, "the cat on the blanket" vs "the blanket on the cat" have the same semantic representation. This pitfall can be avoided by capturing only the head of noun phrase (with adjectives) and instead predicating any prepositional relationships contained within the phrase. We will also explore additional ways to extract structure from language in the form of explicitly stated intra-sentential cause and effect: "Bob loves Susan because she is kind." We will store the extracted information in "Cause-Effect" groups [13].

In addition to improvements of the encoding, we hope to expand our use of LISA. With the availability of computational resources as well as abundant text, auto-encoding LISAese allows for an explosion of analogical modeling. As research progresses, we hope to use auto-coded LISA's mappings and inferences as learning features on a range of traditional NLP tasks. Success in these domains would demonstrate the importance of analogy as a core cognitive capacity and its salience to a host of higher level computational tasks, for example schema generation [23][1], the Story Cloze Task [20, 21], story classification or even provide some insight into the Winograd Schema challenge [16].

ACKNOWLEDGMENTS

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA9550-12-1-0003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon.

The authors would also like to thank Hani Awni and Megan Emigh for their input throughout the process.

REFERENCES

[1] Niranjan Balasubramanian, Stephen Soderland, Oren Etzioni Mausam, and Oren Etzioni. 2013. Generating Coherent Event Schemas at Scale.. In *EMNLP*. 1721–1731.

[2] Tomai E. Forbus K. Klenk M. Dehghani, M. 2008. An Integrated Reasoning Approach to Moral Decision-Making. *AAAI Conference on Artificial Intelligence (AAAI)* Twenty-Third (2008).

[3] Brian Falkenhainer, Kenneth D Forbus, and Dedre Gentner. 1989. The structure-mapping engine: Algorithm and examples. *Artificial intelligence* 41, 1 (1989), 1–63.

[4] Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse Overcomplete Word Vector Representations. *CoRR* abs/1506.02004 (2015). http://arxiv.org/abs/1506.02004
[5] Scott E Friedman, Kenneth D Forbus, and Jason Taylor. Learning and Reasoning with Qualitative Models of Physical Behavior. (????).
[6] Dedre Gentner. 1983. Structure-mapping: A theoretical framework for analogy. *Cognitive science* 7, 2 (1983), 155–170.
[7] Dedre Gentner and Kenneth D Forbus. 2011. Computational models of analogy. *Wiley interdisciplinary reviews: cognitive science* 2, 3 (2011), 266–276.
[8] Mary L Gick and Keith J Holyoak. 1983. Schema induction and analogical transfer. *Cognitive psychology* 15, 1 (1983), 1–38.
[9] Keith J Holyoak and Paul Thagard. 1989. Analogical mapping by constraint satisfaction. *Cognitive science* 13, 3 (1989), 295–355.
[10] Keith James Holyoak and Paul Thagard. 1995. *Mental leaps*.
[11] John E Hummel and Keith J Holyoak. 1997. Distributed representations of structure: a theory of analogical access and mapping. *Psychological Review* 104, 3 (1997), 427.
[12] John E Hummel and Keith J Holyoak. 2003. A symbolic-connectionist theory of relational inference and generalization. *Psychological review* 110, 2 (2003), 220.
[13] John E Hummel and David H Landy. 2009. From analogy to explanation: relaxing the 1: 1 mapping constraint very carefully. In *New Frontiers in Analogy Research: Proceedings of the Second International Conference on Analogy*. New Bulgarian University, 211–221.
[14] John E Hummel, John Licato, and Selmer Bringsjord. 2014. Analogy, explanation, and proof. *Frontiers in human neuroscience* 8 (2014).
[15] Paul Kingsbury and Martha Palmer. 2002. From TreeBank to PropBank.. In *LREC*. Citeseer.
[16] Hector J Levesque, Ernest Davis, and Leora Morgenstern. 2011. The Winograd schema challenge.. In *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, Vol. 46. 47.
[17] Hongjing Lu, Alan L Yuille, Mimi Liljeholm, Patricia W Cheng, and Keith J Holyoak. 2008. Bayesian generic priors for causal learning. *Psychological review* 115, 4 (2008), 955.
[18] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010
[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[20] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. *Proceedings of NAACL HLT, San Diego, California, June. Association for Computational Linguistics* (2016).
[21] Nasrin Mostafazadeh, Lucy Vanderwende, Wen-tau Yih, Pushmeet Kohli, and James Allen. 2016. Story Cloze Evaluator: Vector Space Representation Evaluation by Predicting What Happens Next. *ACL 2016* (2016), 24.
[22] Derek C Penn, Keith J Holyoak, and Daniel J Povinelli. 2008. Darwin's mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences* 31, 02 (2008), 109–130.
[23] Karl Pichotta and Raymond J Mooney. 2016. Statistical Script Learning with Recurrent Neural Networks. *EMNLP 2016* (2016), 11.
[24] V. Punyakanok, D. Roth, and W. Yih. 2008. The Importance of Syntactic Parsing and Inference in Semantic Role Labeling. *Computational Linguistics* 34, 2 (2008). http://cogcomp.cs.illinois.edu/papers/PunyakanokRoYi07.pdf