

© 2019 by Sean A. Wilner. All rights reserved.

NARRATIVE COMPREHENSION THROUGH ANALOGY: A STUDY IN  
COGNITIVE MODELING AND NARRATIVE CLUSTERING

BY

SEAN A. WILNER

DISSERTATION

Submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy in Informatics  
in the Graduate College of the  
University of Illinois at Urbana-Champaign, 2019

Urbana, Illinois

Doctoral Committee:

Professor John Hummel, Chair  
Associate Professor Corina Roxana Girju  
Associate Professor Julia Hockenmaier  
Professor Cynthia Fisher

# Abstract

As the field of natural language processing improves and finds its way into everyday use its current limitations and shortcomings become all the more apparent. The next generation of NLP systems will need to be able to handle tasks at a higher level, drawing together information beyond the lexical and across sentence boundaries. To address this need, research into the field of discourse understanding has emerged as a current hot topic with special attention being drawn to narrative comprehension. We explore cognitive modeling and the application of derived measures of analogy to tasks in the discourse/narrative domains. First, we present improvements to the LISA model, a state-of-the-art cognitive model of analogy, increasing the model's flexibility and robustness, extending the model's functionality to include a probabilistic measure of belief, and presenting an algorithm for automatically producing the model's encoding. Finally we test the utility of narrative analogy as a feature for the Story Cloze Task. We find that narrative analogy is a poor feature on its own, but as part of a composite model with sentiment analysis, it outperforms the best task-given baselines but under-performs state-of-the-art. More importantly, through failure analysis we find that narrative analogy, as conceptualized by the field, is insufficient for such tasks, and researchers must first be able to determine *when* an analogy should be drawn since simply finding all potential analogies proves insufficient.

# Acknowledgments

This dissertation is the culmination and distillation of countless lectures and even more discussions with friends and faculty across a broad scope, without whose support this document would not exist today. Many thanks especially to my adviser, John E. Hummel, who has helped me solidify my theory and guided me through numerous pitfalls in my research. Also thanks to my committee members, Roxana Girju, Julia Hockenmaier, and Cynthia Fisher, who offered me their guidance and support in this project. An additional thanks to International Business Machines (IBM) for awarding me an IBM PhD Fellowship and to the Air Force Research Laboratory's grant funding, which provided me with the financial means to complete this degree. And finally, special thanks to Megan Emigh, Hani Awni, Katherine Wood, Michael Braverman, Pamela Clevinger, John Clevinger, Rachel Flood, Emily Cunningham, Ryan Musa, Rob Deloatch, Jason Rock, Yonatan Bisk, Christos Christodoulopoulos, Philip Miller, Tinkam Ho, Laurie Wilner, Paul Wilner, and numerous other friends and family whose insight and patient ear have helped me refine my thoughts and research directions over the years.

# Table of Contents

<b>List of Tables</b> . . . . .	<b>vii</b>
<b>List of Figures</b> . . . . .	<b>ix</b>
<b>List of Abbreviations</b> . . . . .	<b>x</b>
<b>Chapter 1 Introduction</b> . . . . .	<b>1</b>
<b>Chapter 2 Cognitive Models of Analogy</b> . . . . .	<b>6</b>
2.1 Structure Mapping Engine . . . . .	9
2.1.1 Overview . . . . .	9
2.1.2 Representation . . . . .	10
2.1.3 Recall . . . . .	10
2.1.4 Mapping . . . . .	12
2.1.5 Inference . . . . .	13
2.1.6 Learning . . . . .	13
2.2 Analogical Mapping by Constraint Satisfaction . . . . .	13
2.2.1 Overview . . . . .	13
2.2.2 Representation . . . . .	14
2.2.3 Recall . . . . .	14
2.2.4 Mapping . . . . .	14
2.2.5 Inference . . . . .	16
<b>Chapter 3 LISA</b> . . . . .	<b>18</b>
3.1 Overview of LISA . . . . .	18
3.2 Uses of LISA . . . . .	18
3.3 Constraints/Assumptions . . . . .	19
3.4 Structured Representation . . . . .	20
3.5 Memory Access . . . . .	23
3.6 Mapping . . . . .	24
3.7 Inference . . . . .	28
3.8 Schematization . . . . .	29
3.9 Comparison to Other Models . . . . .	30
3.9.1 SME . . . . .	30
3.9.2 ACME . . . . .	31
<b>Chapter 4 Extensions and Changes to the LISA Model</b> . . . . .	<b>32</b>
4.1 Code-base Changes . . . . .	32
4.2 Multi-way Mapping . . . . .	33
4.2.1 Impact of Multiple Simultaneous Mappings . . . . .	34
4.2.2 Changes to Initialization . . . . .	34
4.2.3 Results of Multi-Way Mapping . . . . .	35

4.3	Hebbian Units . . . . .	36
4.3.1	Calculating Hebbian Consistency . . . . .	36
4.3.2	Changes to Hebbian Consistency . . . . .	37
4.3.3	Hebb Unit Algorithm . . . . .	38
4.3.4	Changes to Hebbian Weight Updating Rule . . . . .	39
4.4	Changes to Unit Activation . . . . .	41
4.4.1	Limits on Activation Aggregation . . . . .	41
4.4.2	Semantic Activation . . . . .	42
4.5	Inductive Confidence . . . . .	43
4.5.1	Implementation of Inductive Confidence . . . . .	43
4.5.2	Tests of Inductive Confidence . . . . .	47
4.6	Conclusion . . . . .	48
<b>Chapter 5</b>	<b>Automation of LISA . . . . .</b>	<b>49</b>
5.1	Representational Requirements . . . . .	49
5.2	Semantic Representation . . . . .	50
5.2.1	Object Semantics . . . . .	51
5.2.2	Predicate Semantics . . . . .	53
5.3	Structural Representation . . . . .	58
5.3.1	Predicate Role Filling . . . . .	59
5.3.2	Narrative Chains . . . . .	62
5.4	Constructing LISAese . . . . .	63
5.5	Simulations . . . . .	64
5.6	Concluding Thoughts and Model Limitations . . . . .	66
<b>Chapter 6</b>	<b>Story Cloze . . . . .</b>	<b>68</b>
6.1	Story Cloze Task . . . . .	68
6.1.1	Origins of the Task . . . . .	69
6.1.2	Previous Approaches to Story Cloze . . . . .	70
6.1.3	Similar Work . . . . .	71
6.1.4	The Dataset . . . . .	72
6.2	Analogical Methodology . . . . .	72
6.2.1	Similarity Function . . . . .	72
6.3	Narrative Analogical Clustering . . . . .	80
6.3.1	Previous Uses of Event Clustering . . . . .	80
6.3.2	Narrative Structure . . . . .	81
6.3.3	Verb Semantics . . . . .	82
6.3.4	Clustering Techniques . . . . .	83
6.3.5	Markovian Clustering . . . . .	86
6.4	Constructing Features . . . . .	88
6.4.1	Clustering for Story Cloze . . . . .	89
6.5	Results . . . . .	90
6.5.1	Follow-up Experimentation . . . . .	93
6.6	Conclusion . . . . .	97
<b>Chapter 7</b>	<b>Contributions . . . . .</b>	<b>99</b>
7.1	Improvements to LISA . . . . .	99
7.1.1	LISA Algorithm . . . . .	100
7.1.2	Confidence Measure . . . . .	101
7.1.3	Automation of Encoding . . . . .	101
7.2	Narrative Analogy as a Feature . . . . .	102
7.2.1	Analogical Similarity Measure . . . . .	102
7.2.2	Discovered Problems With Narrative Analogy . . . . .	103

<b>Appendix A</b>	<b>LISA</b>	<b>104</b>
A.1	SP Unit Firing Order	104
A.2	Unit Activation	105
<b>References</b>		<b>106</b>

# List of Tables

1.1	Story Cloze Task example story with four context sentences and two candidate fifth sentence completions. . . . .	5
4.1	A representative Example of multi-way analogy. With the potential for inference onto the smaller Joan analog to complete the story with Joan’s desire to go to LAX causing her to perform locomotion (mix of semantics between drive_to and fly_to) with the Civic to LAX. . . . .	36
4.2	A multi-way analogy example involving noise, where the ‘prefer’ analog does not match semantically or structurally to the other two stories. . . . .	36
4.3	The Inductive Confidence LISA assigned to the Proposition units in the “Bill” story after considering mappings to and from the “Prefer” and “Joan” stories. . . . .	48
5.1	The cosine similarity scores of the embeddings for “baking”, “frying”, and “canning” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300. . . . .	54
5.2	The cosine similarity scores of the embeddings for “die” vs “expire” and “live” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300 and by Co-reference Chain PMI generated over a corpus of English as a Second Language (ESL) stories using dependencies for verb-roles (bake_nsubj for ‘baker’) where die <sub>PMI</sub> is for the nsubj dependency relation on the verb “die”. . . . .	55
5.3	The cosine similarity scores of the embeddings for “give” vs “donate” and “take” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300 and by Word2Vec embeddings using co-reference chains generated over a corpus of English as a Second Language (ESL) stories using SRL for verb-roles (give_A0 for ‘giver’). Here we see the same effect as with PMI in Table 5.2. We use different verbs show that the same property holds with transitive verbs as well. . . . .	57
6.1	Performance on Story Cloze Task. Choose 1 <sup>st</sup> is shown to reflect that, in the data, the first listed possible 5 <sup>th</sup> sentence is slightly more likely to be correct. . . . .	70
6.2	An example of a potential pairwise similarity matrix on verb-roles. Shaded entries correspond to the selections for an ideal mapping. . . . .	75
6.3	An example of a potential pairwise mapping-scores matrix over narrative chains. Shaded entries correspond to the selections for an ideal object alignment. . . . .	77
6.4	Number of clusters found for different inflation parameters using the three semantic representations given in Subsection 6.3.3. . . . .	90
6.5	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using the sparsified semantics generated from PropBank frames’ role descriptions. . . . .	91
6.6	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using the sparsified semantics generated from co-occurrence in narrative chains. . . . .	91
6.7	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using only the embeddings of the base verb. . . . .	91



6.8	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced with PropBank frames' role descriptions joined with sentiment scores for each sentence. Included are the percentage error reductions over using the sentiment scores alone. . . . .	92
6.9	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using the sparsified semantics generated from co-occurrence in narrative chains joined with sentiment scores for each sentence. Included are the percentage error reductions over using the sentiment scores alone. . . . .	92
6.10	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced with verb-role semantics made using only the embeddings of the base verb. The embeddings are joined with sentiment scores for each sentence to produce a full feature set for each potential story ending. Included are the percentage error reductions over using the sentiment scores alone. . . . .	93
6.11	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using dense semantics generated from PropBank frames' role descriptions. . . . .	94
6.12	Accuracy on the Story Cloze Task using inflation parameters from $i = 2.0$ to $i = 3.0$ with verb-role semantics produced using dense semantics generated from PropBank frames' role descriptions. The embeddings are then joined with sentiment scores for each sentence to produce the final features. Included are the percentage error reductions over using the sentiment scores alone. . . . .	95

# List of Figures

2.1	SME mapping structure for the stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob bakes a cake. The cake cools.” with an added affordance. The dotted lines show the inferred proposition <i>eat</i> (Bob, cake) based upon the discovered mapping between the stories. . . . .	11
2.2	Structure of ACME for the mapping between the stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob baked a cake. The cake cooled.”, the solid lines indicate mutual excitation and dotted lines indicate mutual inhibition. . . . .	15
3.1	LISA representation of the two stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob bakes a cake. The cake cools.” The ovals are Proposition units, rectangles are SP units, diamonds are Object units, hexagons are Predicate units, and the triangles are Semantic units. Thicker lines are used to differentiate higher-weighted semantic edges for easier discussion on the winner-take-all one-to-one mapping preference LISA employs. . . .	21
4.1	The impact to Inductive Confidence decay from current unit confidence. . . . .	46
5.1	The above story has two distinct narrative chains, one following “Bob” and the other following “cake”. Because passive voice subjects are narratively similar to direct objects, we encode them as such. . . . .	55
6.1	To understand mapping between narratives we look at mapping between the chains which comprise them. Every chain is compared to every other chain. Only the outgoing edges from <i>chain1a</i> are represented here to reduce clutter. . . . .	73
6.2	Similar to Figure 6.1, to figure out a mapping score between narrative chains, we breakdown the chains into verb-roles and compare mappings between them, thus reducing the problem to a semantic comparison task. . . . .	75

# List of Abbreviations

ACME	Analogical Constraint Mapping Engine
AI	Artificial Intelligence
AMBR	Associative Memory-Based Reasoning
ARCS	Analogical Retrieval by Constraint Satisfaction
CWSG	Copy With Substitution and Generation
ESL	English as a Second Language
GOFAI	Good Old-Fashioned Artificial Intelligence
IC	Inuctive Confidence
LISA	Learning and Inference with Schemas and Analogies
LSTM	Long Short-Term Memory
LTM	Long Term Memory
MAC/FAC	Many Are Called but Few Are Chosen
MCL	Markov CLuster algorithm
NER	Named Entity Resolution
NLP	Natural Language Processing
NLU	Natural Language Understanding
OP	Object Predicate
PDTB	Penn Discourse TreeBank
PMI	Pointwise Mutual Information
RBF	Radial Basis Function
SME	Structure Mapping Engine
SP	Sub-Proposition
SRL	Semantic Relation Labeling
Word2Vec	Word To Vector
WSJ	Wall Street Journal

# Chapter 1

## Introduction

All communication involves faith; indeed, some [linguists] hold that the potential obstacles to acts of verbal understanding are so many and diverse that it is a minor miracle that they take place at all.

---

*Reason, Faith, & Revolution: Reflections on the God Debate*

- Terry Eagleton

The act of communication between people consists of more than a simple exchange of phonemes or morphemes. Indeed, the actual utterances used to convey meaning are vastly insufficient and under specified. The problem of under-constrained expression extends to syntactic communication and even symbolic logical representations [Green, 2012]. Proper communication requires the ability to properly interpret the material being communicated through context and the lens of personal knowledge. This interpretation is called Natural Language Understanding (NLU), a sub-field of Natural Language Processing (NLP).

With the advent of many NLP systems impacting the every-day lives of most people, it has only become more apparent what the limitations are of the current state of the art. The original promise of home-help and personal assistant AIs are only partially delivered, as conversation and deep context based interpretation remain out of reach. It is as the field pushes up against the boundary of Semantics and Pragmatics that users of NLP systems stop thinking of them as tools and start to wish that the system would *understand* their intent.

NLP, as a whole, has been a hot area of research for quite some time; however, NLU has proven to be far less tractable. A large reason for this is because NLU requires that a system be able to deal with the inherent ambiguities raised by the incompleteness of language where speakers rely heavily upon their audience's interpretation, context, and external knowledge to fill in the unstated information implied in their dialogue. Current trends in NLU research focus on research into pragmatics and discourse, with many specifically centered upon narrative [Lascarides and Asher, 2008, Bex et al., 2007, Tomai and Forbus, 2009, Reiter, 2014]. This trend makes a great deal of sense since narratives play an integral role in human learning and cognition [Tenkasi and Boland, 1993, Rapaport et al., 1989, Landau, 1984]. Unlike many other aspects

of discourse, narratives are often syntactically concrete; in that the boundaries and relations of these inter-sentential constructs can be determined with only intrasentential parsing and agent co-reference resolution [Chambers and Jurafsky, 2008]. However, the question still remains of what to do with narratives once they have been identified. To that end, many researchers have employed a variety of approaches, from combining/comparing narratives to generate frames or schemas [Chambers, 2011, Cheung et al., 2013, Pichotta and Mooney, 2014, Pichotta and Mooney, 2016] to training classifiers over them to categorize stories archetypes [Elson, 2012] to recognizing plot progressions [Goyal et al., 2010] to name a few. The most generalize-able of these focus on schema/frame induction which lends itself well to incorporation into other tasks and general use knowledge bases.

Schemas and frames are structures which are used to categorize information about events (potentially objects as well) such that a collection of examples can be distilled to a singular aggregated form. The idea behind them is that from exposure to multiple instances of sufficiently similar examples a knowledge form can be produced that encodes the generic meaning of some higher order structure from which the examples are drawn. For instance, after having eaten at many restaurants, despite the fact that no two meals were identical, a human agent would likely form a schema around restaurants. In this schema, they would encode information regarding commonplace events and features surrounding restaurants. Using this schema, the agent would be able to make inferences about references they've heard to restaurants as yet un-visited, allowing them to better contextualize and understand conversations regarding these otherwise novel experiences.

Schemas are a concept originally drawn from Philosophy, Psychology, and Cognitive Science (initially Immanuel Kant's term) that attempts to capture established frameworks explaining certain aspects of the world in a way which facilitates interpretation and acquisition of new knowledge. The study of schemas continues to be a source of substantial research. Our use of schemas will be centered on the work of John Hummel [Hummel and Holyoak, 1997, Hummel and Holyoak, 2005, Hummel and Holyoak, 2003, Hummel and Landy, 2009, Hummel et al., 2014, Wilner and Hummel, 2017] which has focused on schema induction as a function of analogical inference wherein schemas are generated and expanded by analogical continuation using individual examples. Following this interpretation allows us a great deal of power since we can now turn the multi-part task of deciding what and how to schematize things into an analogical task.

The term 'analogy' is most often colloquially used in reference to 4-way analogy such as

man : woman :: king : queen

This form of analogy has been studied extensively across a wide scope of fields including Education, Lin-

guistics, Philosophy, Psychology, and Computer Science. Indeed, the task of completing 4-way analogies has been used in NLP as a means of measuring performance of semantic embeddings [Pennington et al., 2014, Demski et al., 2014, McGregor et al., 2016]. However, to use analogy to approach discourse, we need to look at the more general form of analogy where the sources and targets are larger intersentential structures rather than individual words. Specifically, we will focus on drawing analogical similarities between two full stories, called narrative analogy. To better understand this process, we can look at the following two stories:

Bob baked a cake. The cake cooled. He ate the cake.

Sam cooked a steak. The steak rested.

The desired analogy is characterized by finding the best possible mappings between elements of each story: ‘Bob’ → ‘Sam’, ‘cake’ → ‘steak’, ‘bake’ → ‘cook’, and ‘cool’ → ‘rest’. Here ‘best’ means mappings which align elements of the story such that mapped elements share the greatest similarity possible while ensuring that the analogous structures are maintained as closely as possible. For instance, if we were to imagine that ‘cool’ → ‘rest’ and ‘Bob’ → ‘steak’ were the best similarity mappings, we would still not choose them since they are structurally inconsistent since the steak rests, but Bob does not cool.

The ultimate utility of this kind of mapping is several-fold. First, it is innately narrative, and thus a worth exploring for its potential use in other narrative tasks. Second, further research into narrative analogy over large bodies of text may allow for constructed knowledge bases of, or features for, narrative completion; that is, given a story we may be better able to predict what comes next. And third, analogy has been shown to allow for substantial cross-domain knowledge transfer [Klenk and Forbus, 2009, Wang and Yang, 2011, Hinrichs and Forbus, 2011]. Through these kinds of mappings, learned structure from one domain may help to interpret new knowledge and facilitate learning in novel domains, making this an especially appealing potential use case for narrative analogy given the rise of heterogeneous datasets and open-domain tasks.

Using analogy as a basis for higher level cognitive tasks is not a novel concept. Indeed, analogy is widely recognized as an integral part of human cognition, and many consider it to be one of the most important aspect thereof [Gentner, 2010, Hofstadter, 2001, Gentner et al., 2001, Penn et al., 2008]. It is therefore unsurprising that the modeling of analogy has been extensively studied, with focus on both explaining cognitive processes and solving computational tasks [Winston, 1978, Falkenhainer et al., 1989, Forbus et al., 1994, Holyoak and Thagard, 1989]. What has not been as well studied is the making of analogies directly over text. Since all these models work with specialized symbolic or connectionist representations, the production of them from raw text is non-obvious. Nonetheless, it has been previously approached by David Elson who used a model of analogical reasoning, Structure Mapping Engine (SME) [Falkenhainer et al., 1989], to find analogous narrative structures across fables to classify them into fable types with favorable results [Elson,

2012]. However, Elson used hand-encoded representations for these stories and does not provide an approach to automation. Unlike Elson, we have produced a fully autonomous method for examining narrative analogies and a novel analogical measure of narrative similarity.

To use analogy as a tool for higher domain tasks we needed a model or measure of analogy and a way to produce it quickly and autonomously. There has been previous work on textual alignment, such as parse tree sub-tree matching where the model aligns different sub-trees of a complete parse to associate one syntactic structure with another. This technique has proven useful to a variety of domains including machine translation [Eisner, 2003], relation extraction [Zelenko et al., 2003], and sentiment analysis [Hiroshi et al., 2004]. However these models fail to capture any relation or structure which exists between sentences, missing entirely the narrative aspect we hope to make use of. In order to get at analogy, we instead need to keep track of long range intersentential relations, and use them to govern the mappings we discover.

Fortunately we can instead turn to cognitive models of analogy. Specifically, we will look at the Learning and Inference with Schemas and Analogies (LISA) model [Hummel and Holyoak, 1997, Hummel and Holyoak, 2003].

LISA is a psychologically plausible neurally inspired model of human analogy, meaning that it focuses on modeling human performance with cognitively inspired algorithms over a neural architecture. The work described will fall under two main categories: cognitive modeling of analogy & the use of analogy as a feature for general machine learning. Towards those ends, LISA serves as the nexus for our studies. We will explore both expansions to the model itself as well as applications of the model to NLP tasks.

Specifically, with regard to LISA, we altered core components of the model to address issues ranging from mapping errors to optimization along with expanding the model to include a measure of confidence to address complaints levied against the model by others in the field [Gentner and Forbus, 2011]. Furthermore, we developed an architecture for automatically producing encodings usable by LISA to make analogies directly on text. This serves to separate the modeler from the model and allows further research to distinguish exactly what claims are inherent to LISA versus what arises as a result of choices in encoding.

In NLP, narrative analogy has been a feature of interest for several years, appearing mostly in use for what Chatman [Chatman, 1980] would call story based tasks such as fable categorization [Elson, 2012], story generation [Zhu and Ontanón, 2010], or plot progression [Finlayson, 2012]. Here we will explore the potential and pitfalls of using analogy for more general comprehension tasks. Specifically, we will focus on its utility for the Story Cloze Task [Mostafazadeh et al., 2016a, Mostafazadeh et al., 2016c, Mostafazadeh et al., 2017]. The Story Cloze Task consists of taking a short five-sentence story and choosing between two alternative 5<sup>th</sup> sentence completions as shown in Table 1.1.

Story Context	Marcy received a valentine from her boyfriend. It was a card with a gift card in it for chocolates. She was happy and immediately ordered the chocolates. Her boyfriend came over and asked if she liked it.
Story Completions	<i>correct</i> : Marcy gave her boyfriend a big kiss and said yes. <i>incorrect</i> : Marcy shrugged her shoulders and reached for the TV remote.

Table 1.1: Story Cloze Task example story with four context sentences and two candidate fifth sentence completions.

Ultimately, our goals can be summed up into two categories: improvements and extensions to LISA and analyzing the usefulness of narrative analogy for story-based tasks in NLP.

In improving LISA, we will make changes to several of the core components of the algorithm to make it more robust. We will also add in new functionality to the model, allowing it to now address whether a given event is not only possible, but also probable. Lastly, we will present a means to automatically produce LISA’s encodings, allowing for both better analysis of the model’s claims as well as more widespread use of the model.

For analysing the utility of narrative analogy in story-based tasks, we will develop a cheap means to directly measure the analogical similarity between stories for use in narrative clustering and as a feature for the Story Cloze Task. Based upon the results found on the Story Cloze Task, we will examine the limitations of narrative analogy as a feature. Through our failure analysis we will decide that finding “good” analogies is not enough, and that deciding when to draw analogies in the first place is fundamental to their effective use in open-domain story-based tasks like the Story Cloze Task.



## Chapter 2

# Cognitive Models of Analogy

Narrative analogy is a complicated process which remains a widely open topic in both theory and computational modeling. Fundamentally, the task of analogy is to draw a comparison between two structured representations. In practice, it is agreed upon that these representations should include relational, predicated structures between entities and, at times, other relations. This relational nature of representation is at contrast to a simple bag-of-features or an embedded feature vector. The importance of this relational encoding is supported by several studies showing human reliance on relational structure during analogy making [Clement and Gentner, 1991, Gentner and Kurtz, 2006, Spellman and Holyoak, 1992]. Using relational structure, the task of analogy is then to decide in what manner the two structures are related, finding what pieces of one are *analogous* to which pieces of the other. Given the computational difficulty of this problem and the fact that humans readily solve it, since we would like to use narrative analogy on natural language tasks, it therefore behooves us to consider how analogy is performed by humans. To this end we will look into several cognitively inspired models of human analogy. Worth noting is that the goal of these models is not to solve any linguistic, visual, or other fixed computational task. Instead, the following models aim to provide accounts for human performance, including errors, in the task of analogy. Understanding the challenges and solutions proposed by the models can help us to better utilize narrative analogy.

In general, forming an analogy between stories can be decomposed into the following pieces: Representation, Recall, Mapping, Inference, and Learning [Gentner and Forbus, 2011, Gentner, 1989, Holyoak and Thagard, 1989, Hummel and Holyoak, 1997].

1. Representation: When given a narrative, how can we encode it to facilitate productive analogy making?  
This could take the form of abstract embedding into a high dimensional space, or focus more on a structured – often graph or predicate logic based – representation, or sit somewhere in-between the two.
2. Recall: Given a representation, how can we best identify potential targets for analogy making from some set of known stories' representations? This is often accomplished by predicate matching, seman-

tic comparisons, structural similarities, or some combination thereof.

3. Mapping: Mapping is perhaps the most central aspect to analogy. The main question which drives it is: “Given two representations, what is the best coordination (alignment) between the components of each representation?” That is, what elements of one story are analogically similar to which elements of another? More concretely, given a story of a dog chasing a cat and another story of a cat chasing a mouse, is the dog more like the cat or the mouse in the second story?
4. Inference: When one representation contains more information than its analogous representation, how can we (and when should we) extend the analogy to the analogous representation such that the potentially missing information is added? Inference here allows us to both expand narratives to include unmentioned, pragmatically assumed, events as well as hypothesize about likely future events the narrative might contain. Worth noting is that inferences of this type are far from guaranteed. Indeed, analogical inference alone is good at determining what is possible, but insufficient to know what is plausible.
5. Learning: In short, Learning allows the model to move beyond Inference and begin to say what must be, or at least what is likely to be. The actual process of Learning is more open-ended than the other 4 steps of the analogy process. Some models may propose schema induction as a means of “learning”, while others propose identifying statistical scripts, or even constructing structured hierarchical knowledge-bases. What they all share in common is that a good model of analogy needs to have some means of digesting multiple analogies formed between narratives to further some goal of understanding or a meaningful task.

While the accounts for these steps remain hotly debated, there are some core tenants on which many researchers agree, and as such should be kept in mind when devising any means of computational analogy making.

1. Representation cannot be holistic. Narratives are constructed of smaller components, and these components must be represented independent of one another. The components themselves must also be dealt with in a structured fashion rather than as a bag of features (i.e. argument role binding). [Hummel and Holyoak, 2003]
2. Recall is largely influenced by surface level features rather than structural components, so a story including a dog chasing a cat is similarly likely to be retrieved as the same story replaced with a cat chasing a dog. [Ross, 1989, Gentner et al., 1993, Holyoak and Koh, 1987]

3. Mapping is one-to-one, type restricted, and structurally consistent. Here one-to-one mapping means that no two entities should be mapped onto the same entity. Type restriction is the requirement that only like types will map onto one another, for instance objects cannot map onto predicates. Lastly, structural consistency is the requirement that if two units map onto one another, their parent units must as well. For example, given that two predicates map onto each other, the propositions to which they belong must likewise map onto each other. [Hummel and Holyoak, 2003, Hummel and Holyoak, 1997, Gentner, 1983, Gentner and Markman, 2006, Keane et al., 1994]
4. Inference preserves entity mapping and inferred propositions are generated from the source of the analogical inference by inducing similar or identical structure and filling them with predicates and objects either already present in the target analog or inferred during this inference as dictated by preservation of mapping consistency. This is called Copy With Substitution and Generation (CWSG). [Hummel and Holyoak, 2003, Holyoak and Morrison, 2005]
5. Unfortunately, approaches to learning are so broad that there is little to no universal common ground [Gentner and Forbus, 2011]. Indeed, one of the few things agreed upon is the premise itself; that is, that the process of analogy can facilitate learning [Gick and Holyoak, 1983, Catrambone and Holyoak, 1989].

Understanding the constituent pieces of analogy, we can address whether analogical modeling as defined is meaningful. Fundamentally, the question at hand is whether analogy of the type described above is a purely synthetic task, or if it actually has some basis in human cognition. Certainly, the *use* of analogy includes more pieces than are covered here, most notably when and how to use discovered analogies, but there is evidence that actual process of finding the analogies between stories and human cognitive behavior is relatively well modeled by this paradigm [Morrison et al., 2004, Viskontas et al., 2004, Markman and Gentner, 2000, Larkey and Love, 2003]. While these studies show that computational models of analogy based upon the five elements listed above can account for human performance on fixed analogy tasks, the tasks themselves are not entirely natural. Fundamentally, analogy as described above, while perhaps somewhat artificial in construction, does seem to accurately represent at least a part of the analogical processes we as humans use to help us understand and interpret stories. As such, while not complete, the models built upon these five elements are useful and worthwhile for continued study to the extent that they can help us, even in part, unravel the approach people use for the task of narrative analogy.

As we look into models of analogy, we will examine the strengths and weaknesses of how these tasks are approached. There are a wealth of models of analogy [Kokinov, 1988, Kokinov and Petrov, 2001, Hofstadter

et al., 1994, Mitchell, 1993, Doumas et al., 2008] which can roughly be divided into three main structural types: Symbolic, Connectionist, & Hybrid. Symbolic models are ones where the structural elements can be combined to represent meaning, and when re-combined in different configurations (or with different elements) can represent new and different meaning. Symbolic models are often represented with predicate logical notation, and the category is often called “Good-Old-Fashion-AI” or GOFAI and contains most rule-based, expert systems. Connectionist architectures on the other hand operate in a neural-network-like fashion, passing along activation and inhibition between structural units. The Hybrid class of models operate, at least in part, with a Connectionist activation-passing, but also maintain Symbolic units. To avoid competing architectural requirements, in practice Hybrid models are often broken apart into different levels, on which they operate either in a Symbolic or Connectionist fashion independently of the other levels. Here we will focus our attention here on three models of special impact on the field:

- The Symbolic model SME [Gentner and Markman, 1997]
- The Connectionist model ACME [Holyoak and Thagard, 1989]
- The Hybrid model LISA [Hummel and Holyoak, 1997]

## 2.1 Structure Mapping Engine

Structure Mapping Engine, or SME, is one of the most prolific computational models of analogy, giving rise to several companion models including most notably MAC/FAC for managing recall [Forbus et al., 1995] and SEQL for handling generalization/learning [Kuehne et al., 2000]. SME’s influence extends beyond cognitive modeling and into computational solutions to various tasks, including fable classification [Elson, 2012], poetry generation [Manurung, 2004], or semantic network generation [Baydin et al., 2015] to name a few.

### 2.1.1 Overview

SME uses sub-graph matching to align the largest pieces of potential analogs to establish an analogy. While this task (maximum induced sub-graph matching) is in general NP-complete [Kann, 1992, Garey and Johnson, 2002], SME constrains it by assigning each node in the graph a specific type, and restricting the matching to be type-preserving. SME then further restricts the search space by only mapping predicates onto identical predicates. The representation SME uses to form its graphs is based on a predicate calculus-like representation [Gentner, 1983, Gentner, 1989, Holyoak and Morrison, 2005].

### 2.1.2 Representation

SME’s representation is a symbolic graph encoding based upon the logical form of a statement where graph edges are listed left to right to indicate the order of the predicate attachment. For instance, in the upper half of Figure 2.1 we can see SME’s representation of the story “Sam cooks a steak. The steak rests. Sam eats the steak.”

Here we can see SME’s graph-based representation broken apart into SME’s four types: Objects, Attributes, 1<sup>st</sup> Order n-ary Relations, and Higher Order n-ary Relations.

- Objects refer to both active agents as well as inanimate objects such as either “Sam” or “steak”.
- Attributes are single place predicates like “rest” which take a single object like “steak” as its only argument. SME does not typically bother to represent these units unless they participate in a higher order relation since there is no structural information conveyed by a single place relation.
- 1<sup>st</sup> Order n-ary Relations actually refers to a class of types where each ‘n’ corresponds to a different cardinality of the predicates (i.e. a predicate which takes ‘n’ roles). For instance, “cook” is a 1<sup>st</sup> Order binary relation, which takes two objects, “Sam” and “steak”. In fact, attributes are a special case of 1<sup>st</sup> order unary relations.
- Higher Order n-ary Relations likewise refers to a class of relations which in turn take lower order n-ary relations as their arguments. There are several types of common higher order relations which one might want to represent, such as causal links or affordances. For example, in Figure 2.1 we include the higher order relation *afford(rest, eat)* to represent that the steak resting affords Sam eating it.

### 2.1.3 Recall

SME does not handle recall itself, but rather relies upon the associated companion model “Many Are Called but Few Are Chosen”, or MAC/FAC [Forbus et al., 1995], to perform recall for it. MAC/FAC works in two stages, in the first, the MAC, a “content vector” which counts the number of times any given predicate appears in the analog is constructed (the length of this vector is the same as the length of all known predicates). This vector is then compared to all other such vectors for analogs in memory, and the best 10% are selected to be passed to the second stage, FAC. FAC then fully runs SME to select the best candidates.

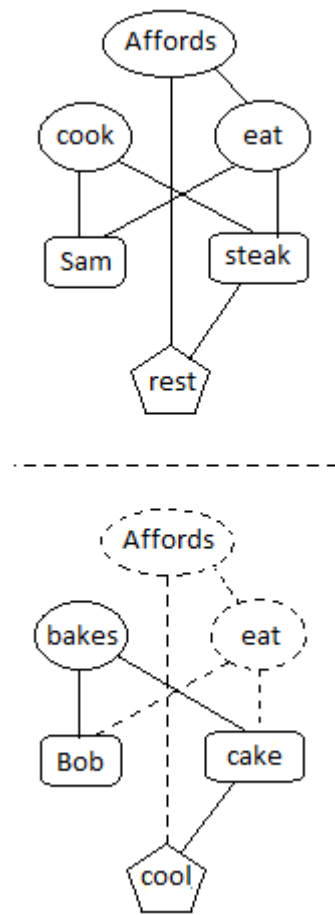


Figure 2.1: SME mapping structure for the stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob bakes a cake. The cake cools.” with an added affordance. The dotted lines show the inferred proposition *eat*(Bob, cake) based upon the discovered mapping between the stories.

### 2.1.4 Mapping

Mapping in SME works by first restricting potential sub-graph matches such that predicates are mapped to identical predicates. Then the potential mappings are sorted through to find the largest potential sub-graph match. If no such mapping is found, the predicates go through what Genter et al. call “re-representation” whereby predicates are re-written into a similar hyponymous word.

To see this in action, we can look again to Figure 2.1. Initially, SME is provided with the input as shown in the Figure, less those made up of dotted lines. A given story is set as the source while the other is the target for analogy discovery. For our example, let us say that the top story is our source and the bottom our target. So, SME looks for identical predicates between the source and target. Finding the top story contains ‘cook’, ‘rest’, ‘eat’, and ‘Affords’ and the bottom story has ‘bakes’ and ‘cool’, SME finds no matching predicates. This triggers SME’s “re-representation” where predicates are raised to a higher ontological level (made a more general version of the same event). In the case of our example, we might find that both ‘cook’ and ‘bake’ raise-up to ‘prepare’ and so SME would re-represent the two stories as:

“Sam prepared a steak. The steak rested. Sam ate the steak.”

“Bob prepared a cake. The cake cooled.”

Armed with these new stories, SME can find the identical predicate match on ‘prepared’ and extend the structural similarity as much as possible, with the left-most actor of one of the ‘prepare’s mapping onto the left-most actor of the other. This yields a mapping from ‘Sam’ to ‘Bob’ and, by extension to the right-most actors, ‘steak’ to ‘cake’. SME now tries to match the remaining relations and objects in the source (attributes do not drive mapping except through attachment to a higher-order relation such as ‘Affords’). Since there are no un-mapped objects left, we are only left with the relations ‘eat’ and ‘Affords’. The simplest of these to resolve is ‘Affords’. Since ‘Affords’ is a second order relation, and since the representation of the bottom story has no second order relations whatsoever, SME cannot possibly find a matching unit and marks it down as *un-mapped*. Next, trying to find a mapping for ‘eat’ SME looks for a predicate that can be re-represented to match ‘eat’. Since no such predicate exists in the bottom story (recall that the dotted pieces of the Figure do not yet exist as they represent inferred structure), SME marks down the predicate ‘eat’ as *un-mapped*.

Functionally, once SME finds matching predicates (or re-represented matching predicates) the task of mapping is reduced to finding the largest possible matching sub-graph of each story’s representation such that matched relation nodes are identical (or re-represent-able to be identical).

### 2.1.5 Inference

After SME has discovered a mapping, if either analog has nodes not contained in the matched sub-graph, those nodes can be inferred in the other analog. SME does this by extending the matched sub-graph, creating new nodes to represent the missing elements. For instance in Figure 2.1 “Affords” and “eat” are inferred in the lower graph by extending it with nodes from the upper graph to expand the sub-graph match to encompass the whole analog. In this process, SME would then look to re-represent ‘cool’ and ‘rest’ to the same attribute, perhaps ‘reduce-entropy’ to avoid constructing new attribute units in the lower story.

These inferred units are produced from a single extension, and as such can hardly be relied upon with any surety. To establish how confident a model is in its generated inferences we must consider the inferred units across multiple instances of the same or similar stories. The collation of such inferences falls under the category of Learning.

### 2.1.6 Learning

Like recall, SME itself does not perform any learning and instead relies upon a companion model called SEQL [Skorstad et al., 1988, Kuehne et al., 2000]. In SEQL learning takes place in the form of generalizing analogs into statistical schema abstractions when sufficient numbers of analogous exemplars are found. The generalization takes place by abstracting the elements of each analog until all the subgraph matches are identical. SEQL then stores the conjunction of those along with a score on how frequently it finds each element in the exemplars.

## 2.2 Analogical Mapping by Constraint Satisfaction

Analogical Mapping by Constraint Satisfaction, or ACME, is a model of analogy which uses parallel constraint satisfaction to calculate potential mappings. It enforces structural constraints with a preference for notional similarity between mapped elements where the definition of similarity is left up to the modeler.

ACME has been a model of significant importance to the development of the field, heavily influencing several prominent models including LISA [Hummel and Holyoak, 1997].

### 2.2.1 Overview

ACME uses a structured, predicated representation reminiscent of SME’s. However, where it differs from SME is that rather than focusing on predominantly structural constraints, instead it attempts to satisfy structural, semantic, and pragmatic constraints all at the same time. Since ACME performs its mapping over



several restrictions, another difference from SME is that any of these constraints (including structural) can be violated for a given mapping so long as the local violation improves the global adherence to the constraints as a whole.

### 2.2.2 Representation

ACME's representation uses a connectionist representation unlike SME. Rather than having nodes in our graph represent the individual elements, ACME instead uses them to represent potential mappings. These mapping-nodes are connected to other mapping-nodes and to nodes representing pragmatic importance and semantic similarity of the mapped entities, labeled 'Purpose' and 'Similarity' in Figure 2.2.

In Figure 2.2, the analogy being modeled is on the narratives: "Sam cooked a steak. The steak rested. Sam ate the steak." and "Bob baked a cake. The cake cooled." In logical notation:  $\text{cook}(\text{Sam}, \text{Steak}) - \text{rest}(\text{steak}) - \text{eat}(\text{Sam}, \text{steak})$  v.s.  $\text{bake}(\text{Bob}, \text{cake}) - \text{cool}(\text{cake})$ . The notion of semantic similarity is encapsulated by the respective weight each mapping gets to the node representing similarity. Similarly, the pragmatic importance of a given mapping is represented by the weight to the corresponding pragmatic node.

### 2.2.3 Recall

ACME itself does not support any recall functionality, however an associated model, Analog Retrieval by Constraint Satisfaction (ARCS) [Thagard et al., 1990] performs recall in an ACME informed fashion.

ARCS handles recall by connecting stored analogs to a semantic network which links similar concepts. In this way, when a new analog is presented, analogs containing similar concepts are proposed as candidates for recall. Once proposed, ARCS begins to perform ACME like constraint satisfaction restricted to those elements of the analogs which are connected to similar concepts in the semantic network. The effect of this is that the recall ARCS performs is more sensitive to semantic match than to structural matches. However, if two candidate analogs share a similar degree of semantic overlap with the source of the recall, the structural differences between them begin to play a larger role in the recall since the constraint satisfaction has the same "amount" of similar concepts at play, but even though it's restricted to only the semantically similar entities, the differences in structure can play a differentiating role, allowing for the recall of potentially better analogs.

### 2.2.4 Mapping

Mapping in ACME takes place by activation settling over the network with activation being driven by the pragmatic and semantic similarity nodes. In our example Figure 2.2 this would begin by feeding activation to the 'Purpose' and 'Similarity' nodes, and allowing them to spread activation until the network reaches

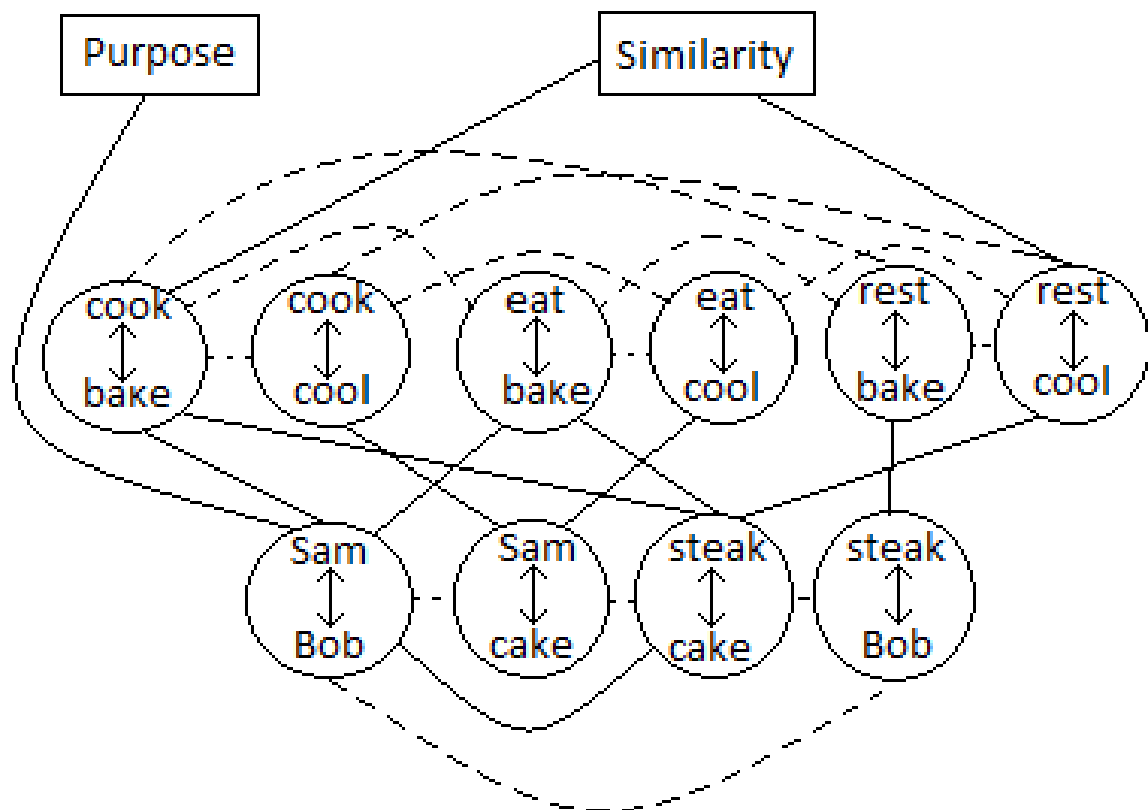


Figure 2.2: Structure of ACME for the mapping between the stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob baked a cake. The cake cooled.”, the solid lines indicate mutual excitation and dotted lines indicate mutual inhibition.

a stable state of activation. Since ‘Purpose’ is only connected to the  $Sam \leftrightarrow Bob$  node, it will activate with excitation from ‘Purpose’ (based upon the weight of its connecting edge). Similarly  $cook \leftrightarrow bake$  and  $rest \leftrightarrow cool$  will activate as well based upon their respective edge weights to the ‘Similarity’ node. By passing activation and inhibition from these nodes along the solid and dotted lines respectively, we find that the only nodes receiving activation are  $steak \leftrightarrow cake$  and  $eat \leftrightarrow bake$  while all other nodes are inhibited. Since  $steak \leftrightarrow cake$  receives activation from all the active nodes and no inhibition (since all its inhibitory connections are inhibited, and thus not active) it activates. However, while  $eat \leftrightarrow bake$  receives activation from  $Sam \leftrightarrow Bob$ , it is inhibited by  $cook \leftrightarrow bake$  and thus is not guaranteed to have a positive activation. Indeed,  $eat \leftrightarrow bake$  and  $cook \leftrightarrow bake$  are in a winner-take-all battle where only one can be active if the model is to maintain structural consistency. Whether the network can settle with both of these units active depends upon the excitatory/inhibitory weights chosen (a model parameter), but in the case that weights are chosen to strongly preference structural consistency – a common practice – one of them will win. In that event, since  $cook \leftrightarrow bake$  receives all the excitatory input that  $eat \leftrightarrow bake$  does plus the input from the ‘Similarity’ node, it will win, and the network will settle. At the time of settling, the active nodes, and thereby the discovered mapping, will be  $cook \leftrightarrow bake$ ,  $Sam \leftrightarrow Bob$ ,  $rest \leftrightarrow cool$ , and  $steak \leftrightarrow cake$  successfully finding the correct mapping.

### 2.2.5 Inference

Inference in ACME follows the CWSG paradigm, whereby un-mapped elements are copied and their arguments are filled in to maintain the consistency of already discovered mappings. That is, if a predicate in the source analog doesn’t map onto the target analog, ACME will copy it over and fill its roles such that if an argument of the predicate in the source maps onto a given element in the target analog, that mapped element will be substituted as the corresponding argument of the copied predicate. Whatever arguments are left (i.e. arguments that do not have a mapping into the target) are then generated in the target analog to fill the remaining empty roles.

To provide a concrete example, in our example shown in Figure 2.2 neither  $eat \leftrightarrow bake$  nor  $eat \leftrightarrow cool$  were active upon settling. In its inference step ACME would find that when mapping is complete, since  $eat(Sam, steak)$  is un-mapped, it is a potential source for inference and begin CWSG. The predicate “eat” would then be copied over to the target analog in the copy step. For substitution, since the mappings  $Sam \leftrightarrow Bob$  and  $steak \leftrightarrow cake$  were discovered, ACME would substitute “Bob” for “Sam” and “cake” for “steak” in the copied “eat” predicate. Ultimately this would generate the inferred proposition  $eat(Bob, cake)$ . As mentioned at the beginning of this Chapter, on their own, inferences like this should be interpreted

as “it wouldn’t be surprising if Bob ate the cake” rather than “I expect Bob to eat the cake”. To get to the latter, several analogies between food preparation themed stories would need to be explored and their core commonalities distilled.

# Chapter 3

## LISA

Learning and Inference with Schemas and Analogies, or LISA [Hummel and Holyoak, 1997, Hummel and Holyoak, 2003], is a neurally plausible cognitive model of analogy<sup>1</sup>. In this chapter, we will provide a high level overview of the LISA model of analogy since the following chapters will rely on a basic understanding of the model’s driving constraints and the approaches used to satisfy them. Here we will endeavor to provide the necessary details while stopping short of the actual implementational minutiae. For a deeper look at the technicalities of LISA, see [Hummel and Holyoak, 1997, Hummel and Holyoak, 2003]

### 3.1 Overview of LISA

LISA is a hybrid symbolic connectionist model of analogy in that it consists of nodes representing both specific lexical and logical atoms which can be recomposed into different configurations to encode different knowledge structures as well as a distributed connectionist semantic representation of universal primitives. LISA makes use of temporal synchrony of firing to associate otherwise unconnected elements of the connectionist architecture, thereby facilitating the analogy discovery process. The meaning of the nodes in LISA is fixed irrespective of the way that they are combined. For example, in LISA, using the predicate “chase”, in *chase(dog, cat)* the units corresponding to “dog”, “cat”, and the roles and representation of “chase” are identical to those used in *chase(cat, dog)*. The difference lies in how the predicate roles and objects are connected with one another. This symbolic structure is called ‘compositional connectionism’ which is required for ‘dynamic binding’, the basis for the mapping which LISA performs [Hummel et al., 2004].

### 3.2 Uses of LISA

To understand LISA’s importance to the field, it is useful to see how the model has been used. Since LISA’s impact has been wide spread with over 1800 citations, we cannot explore all the uses of the model, so we

---

<sup>1</sup>LISA includes a working memory capacity, resolving all mapping computations locally. Several of LISA’s claims have not only been shown to be neurally plausible, but experiments of neural spike timing have shown that LISA’s principal method of establishing coherence and mapping discovery using neural synchrony actually occurs in the human brain.

will instead focus on two especially interesting genres.

The first type we will look at uses LISA to help explain difficulties in analogy making as a result of cognitive decline. Morrison et al. used LISA to model analogy production in patients with frontotemporal lobar degeneration [Morrison et al., 2004]. To model the difference between frontal lobe and temporal lobe degeneration for analogical performance, the authors removed portions of semantic grounding from LISA’s representation, changing LISA’s model parameters for learning rates and lateral inhibition respectively. The authors found that LISA’s performance differed in absolute from human performance, but followed trends nearly identically across the two cases (a fixed difference between model performance and human performance for each case). In a similar vein, Viskontas et al. looked at how age based cognitive decline effects analogy making [Viskontas et al., 2004]. Their theory was that age-based decline in analogical reasoning was driven by a reduction in attention and inhibitory control. By changing LISA’s parameters to reflect these changes, they found that they could account for a full 96% of their data’s variance. The success of these two studies not only suggests that their hypotheses are likely correct, but also provides strong evidence for LISA as a whole, especially since the model was never designed to model cognitive decline in the first place.

The second use case for LISA we will explore centers on functional applications. Tinker et al. used a streamlined, more task specific implementation of LISA’s architecture to predict the success of satellite task scheduling [Tinker et al., 2005]. The process of task scheduling for satellites often takes prohibitively long<sup>2</sup>, but the authors observed an over 80% accuracy in predicting whether a given task would successfully be scheduled for a given satellite within seconds (sub one second for many of their experimental setups). While this application suggests little about LISA’s import to cognitive modeling, it does hold promise for the use of LISA on difficult real-world tasks.

As mentioned above, LISA’s successful application to such a wide variety of tasks for which it was not designed speaks to the robustness of the model and its impact to interdisciplinary science at large.

### 3.3 Constraints/Assumptions

In order to better understand how LISA tackles the challenges of analogy, it is important to understand what assumptions guided the development of LISA. There are five main constraints which LISA assumes must be followed to properly perform analogy:

1. Role Binding: The predication of arguments cannot be treated as a bag of objects. That is, each argument must be bound to a specific role. *chase(dog, cat)* is not the same as *chase(cat, dog)* because

---

<sup>2</sup>The authors place the time-frame at hours, presumably using the hardware which HRL Laboratories, NASA Ames Research Center, and Raytheon Company used at the time of publication since the authors are all from said institutions.

“dog” and “cat” are bound to different roles in the two different predications.

2. Role/Filler Independence: The representation of predicates should be broken apart by their role to facilitate compositional semantics. Whatever representations are used for the roles and fillers should be independent of one another. For example, in the representation of  $chase(dog, cat)$  the representation of both  $chase\_A2$  – the role of the thing being chased – and  $cat$  should be same as the representation for  $chase\_A2$  and  $cat$  in  $chase(cat, mouse)$  even though in the second one the “cat” is the role of the “chaser” instead.

3. Structure Driven Mapping: Good analogical mappings should preserve the relational structure between mapped elements. For instance, given the two stories

The dog chased the cat. The cat hid from the dog.

The cat chased the mouse. The mouse hid from the cat.

we know that mapping  $cat \rightarrow mouse$  yields a better analogy than  $cat \rightarrow cat$  despite the favorable semantic match between the two instance of  $cat$ .

4. Inference by Extension: Inference in analogy happens by extending the discovered mapping to generate a structure which would have resulted in a complete mapping. This constraint is nearly exactly explained by CWSG. For example, if we have two stories such as  $chase(dog, cat)$  &  $scare(dog, cat)$  and  $chase(cat, mouse)$  with the discovered analogy  $dog \rightarrow cat$ ,  $cat \rightarrow mouse$ , and  $chase \rightarrow chase$  we would apply CWSG to extend it and complete the mapping. Since  $scare(dog, cat)$  is un-mapped, CWSG would infer new structures in the second story to allow for all elements to successfully map such as  $scare(cat, mouse)$ .

5. Semantic Grounding: Elements of the structure should have their meaning grounded in some set of semantic primitives. An object like  $cat$  should have some collection of nodes which corresponds to a distributed semantic representation where each of those nodes are connected to the symbolic  $cat$  node. Perhaps these semantic units could be ontological in status such as  $is\_feline$ ,  $kind\_of\_pet$ , or  $has\_claws$ , or they could instead represent a semantic embedding with nodes corresponding to vector dimensions.

### 3.4 Structured Representation

At a high level description, the way that LISA handles knowledge representation is suggestive of a logical predicate calculus representation. Indeed, the scripted out files from which LISA reads its input, called

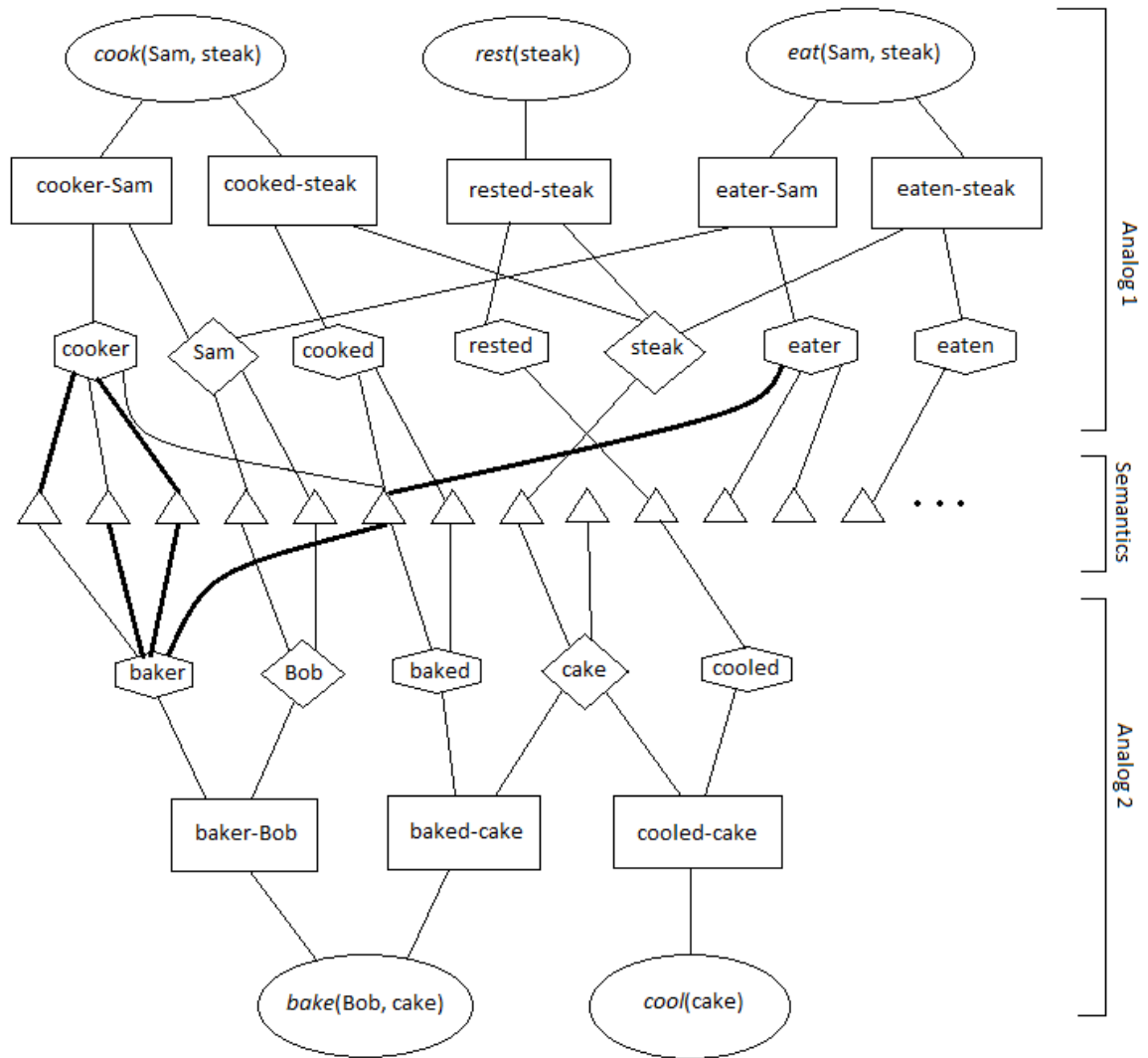


Figure 3.1: LISA representation of the two stories “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob bakes a cake. The cake cools.” The ovals are Proposition units, rectangles are SP units, diamonds are Object units, hexagons are Predicate units, and the triangles are Semantic units. Thicker lines are used to differentiate higher-weighted semantic edges for easier discussion on the winner-take-all one-to-one mapping preference LISA employs.



SYM-files, are written in a predicate calculus like fashion, referred to as LISAese. However, in implementation, it becomes much less obvious that LISA is utilizing anything resembling predicate calculus. LISA's connectionist network employs a structure that, on simple examples, is reminiscent of a forest of tree graphs to represent any given analog. The propositions in the represented analog become the roots of the trees<sup>3</sup>, and the leaf nodes are then semantic units which handle grounding the other symbolic nodes. This description is actually an over-simplification since the semantic units are universal, in that any two objects (either in the same analog, or across analogs) which share at least some semantic overlap, must then connect to the same semantic node(s) (and thus are not technically distinct trees, or even a single tree if objects in the same proposition contain semantic overlap). In practice, it may prove easier to visualize them as trees whose leaf nodes' activation are yolked to other leaf nodes.

To make LISA's structure easier to understand, we will talk about it in a concrete, grounded, fashion by exploring what the LISA representation for following two stories as shown in Figure 3.1:

“Sam cooked a steak. The steak rested. Sam ate the steak.”

“Bob bakes a cake. The cake cools.”

LISA makes use of four levels of representation for each proposition in both stories:

1. Proposition Level: The symbolic representation of the complete logical connection of the proposition. Ultimately, it is this level which encodes the knowledge being represented. In Figure 3.1, these are the ovals sitting at the top and bottom of the Figure. The only instance where this is violated is when LISA is representing a recursive structure where a Proposition unit sits at the Object level. For example, in the sentence “Bob knows that Bill loves Mary.” we have the Proposition *loves(Bill, Mary)* as an argument to the Predicate *knows*, such that the Proposition unit for *knows* would represent the logical predication *knows(Bob, loves(Bill, Mary))*. To distinguish these uses of Proposition units, LISA keeps track of how each individual Proposition unit is operating with a binary switch marking whether the Proposition is being used as a parent or child in a given context.
2. Sub-Proposition Level: Sub-Proposition Units, or SP Units, are the binding glue which distinguishes between different ordering of objects in a give relation; for example, SPs are the way that LISA distinguishes between *loves(Bill, Mary)* and *loves(Mary, Bill)*. These units are necessary, because, unlike SME, LISA's network of nodes does not maintain a fixed geometry (ordering is meaningless in LISA). SP Units represent a substantial divergence from other models of analogy as they allow for

---

<sup>3</sup>This is not strictly the case, as when what LISA calls *groups* are involved they sit at a higher level in the tree than propositions. However, as we will not be covering *groups* here, for comprehension and simplicity's sake we will treat propositions as the roots of LISA's trees. For an understanding of LISA's *group* structure, see [Hummel et al., 2008]

some of the benefits of conjunctive encoding, where the meaning of objects with respect to a proposition is tied to the roles that they fill, while also allowing for the flexibility of independent symbolic representations such that the meanings represented are changed when the units are re-combined into different configurations. In Figure 3.1 SPs are represented as rectangles such as “cooker-Sam”.

3. Object Predicate Level: Object Predicate (OP) units are symbolic representations of concepts at the lexical and predicate-role levels. That is, they are single units which correspond to the words (or roles of words) which would be used to describe the elements of the proposition. For example, in Figure 3.1, the diamonds and hexagons, such as “Bob” and “baker” are OP level units; here “Bob” is straightforwardly a lexical component of the proposition. Similarly, “baker” here represents the agent-role of the lexical verb “bake”, rather than labeling the unit “baker” other possible alternatives would be “*bake\_nsubj*”, “*bake\_agent*”, or “*bake\_A0*” depending upon the modeler’s choice of parsed role-type.
4. Semantic Level: Semantic units are a universal set of nodes (all Analogs access the same Semantic units) which encapsulate the meaning of Object-Predicate Units in a distributed fashion. Historically, in LISA these units have represented either ontological properties, or various aspects of the OP units, such as formal, constitutive, telic, or agentive roles. However, in our further work, these units will represent dimensions of a vector space in which objects and/or verb-roles have been semantically embedded.

Finally, using these four levels of representation LISA represents entire stores, or analogs over which the algorithm runs.

### 3.5 Memory Access

LISA simulates analog retrieval from Long Term Memory (LTM) by splitting up its analogs into a single *driver* analog, a set of *recipient* analogs, and a remaining set of LTM analogs. The *driver* is put into what LISA calls “active memory” which is a set of memory distinct from the “working memory” occupied by *recipient* analogs with a lower threshold to access working memory than LTM. LISA then asynchronously brings each proposition in the *driver* into working memory, activating them one at a time through a time series of activations. Activating the propositions here means firing each proposition, SP, and Object such that they are in synchrony with each of their parent nodes (in the tree structure), but out of synchrony with all their sibling nodes. To clarify, this results in units residing higher in the tree (closer to the proposition

nodes) having longer phases of activation to remain in synchrony with all their children nodes which fire out of synchrony with one another. During each firing, LISA attempts to settle activation before moving on to the next asynchronous activation. For example, when two SP Units belonging to the same Proposition unit fire out of synchrony with one another, their joint Proposition unit will remain active through both of their activations, which only switch activation with each other after the *driver* and *recipients* have had time to settle their respective activations. When the Object Units fire, they activate their corresponding Semantic units. Since Semantic units are universal in LISA, these units then become the connection through which the *driver* activates the *recipients*. When the units in the *recipient* analogs accumulate enough activation to exceed some fixed threshold they are then recalled into working memory, signaling LISA to target them for mapping potential, thereby completing the long term memory access.

### 3.6 Mapping

Mapping discovery constitutes the core work of analogy making in LISA. The process of mapping in LISA is substantially distinct from other models as it relies on synchronous firing to establish connections in a Hebbian learning [Hebb, 1949, Hebb, 2005, Caporale and Dan, 2008] fashion where units which are associated in a “fire together, wire together” type algorithm.

The manner in which LISA maps analogs shares some similarity with the way it handles memory recall. In mapping, LISA asynchronously activates the propositions in a *driver* analog by passing activation to SP Units such that, like in recall, units belonging to the same analog at the same level of the hierarchy are out of synchrony with one another but in synchrony with their parent units, ultimately passing activation to the semantic units attached to the Object and Predicate Units. Then, the *recipients*, again like in recall, are driven, in a bottom-up fashion, by the Semantic units.

To find ideal mappings, LISA cannot rely solely on semantic activations to drive Hebbian learning. In order to find synchronous firings which preference structural consistency in addition to the semantic similarity, LISA combines bottom-up activation with mutual lateral-inhibition between competing units at the same level of the hierarchy (siblings in the tree structure) and excitation passed top-down from connected parent nodes of the tree structure. In order to discover better mappings, LISA allows unconstrained semantic to accumulate first before then restricting activation with lateral-inhibition and only introduces top-down activation towards the end of the mapping discovery process.

To keep track of which units fire together, LISA employs units called “Hebb Units” which, similar to ACME’s structure, represent a mapping between two units of the same type by keeping an activation aggre-

gator that updates on each activation when both units are active. All units besides Semantic units (which are universal, and thus don't map), have a Hebb Unit for each potential mapping to any similarly typed unit belonging to another analog which has been recalled into active memory. LISA then determines which Hebb Units support each other and which oppose each other (i.e. represent mappings which are structurally consistent or inconsistent with each other). Mutually consistent Hebb Units pass activation to each other, while inconsistent ones inhibit one another. After activation has settled, LISA then goes through a period where these Hebb Units then update their weights based upon the mutual activation discovered and the start weight of themselves and the other consistent and inconsistent Hebb Units. Once all asynchronous activations have been run, the Hebb Units with the largest weights correspond to the discovered mappings.

Finally, LISA calculates a score of mapping quality from the *driver* to each *recipient* analog which takes into account the maximum weight of the Hebb Units connected to each unit in the *driver* amortized over the number of units in the *driver* (the sum of best weights corresponding to the possible map of each element of the *driver* divided by the number of such weights). With this score, LISA can determine how analogous it believes two narratives to be.

To see this all in practice, we can observe Figure 3.1 to follow the mapping of the *driver*, Analog 1, onto the *target*, Analog 2. While running, LISA fixes a firing order of what it calls 'phase-sets' which constitute the collection of Proposition units in the *driver* which LISA will consider all together when forming a mapping (if there are structural considerations which only arise when considering multiple propositions at the same time, those propositions must be all included in the same phase-set to find the correct mapping). Here, for simplicity's sake, we will use phase-sets consisting of single Proposition units following the order: *cook*(Sam, steak)  $\rightarrow$  *rest*(steak)  $\rightarrow$  *eat*(Sam, steak). To begin with, LISA will determine which SP unit belonging to *cook*(Sam, steak) to fire first, either "cooker-Sam" or "cooked-steak". The way LISA chooses which SP unit in the *driver* gets to fire in general is determined by the algorithm discussed in Subsection A.1 in Appendix A. For the first choice however, the random noise in Equation A.1 handles breaking the symmetry between the two SP units, making the initial SP unit activation a random selection. Let us say that LISA settles on activating "cooker-Sam" first. The "cooker-Sam" unit will pass activation along its connected edges to *cook*(Sam, steak), "cooker", and "Sam". As the input to these units grows, so too does their respective activations, allowing them to in turn activate their connected units. During this step, all SP units other than "cooker-Sam" are strongly inhibited, and so the activation from *cook*(Sam, steak) to "cooked-steak" and from "Sam" to "eater-Sam" does not produce a positive input in the corresponding SP units. However, there is no such inhibition on the Semantic units attached to "cooker" and "Sam" which will become active. This semantic activation will in turn provide input to "baker", "Bob", and "baked" in Analog 2 as they are all

attached to active semantic units. LISA aggregates input of the units and their activation will ramp up to meet their input, as described in Subsection A.2 in Appendix A, eventually passing activation to their parent SP units and Proposition units. During the first portion of a phase-set, LISA does not restrict activation in the *recipient* and both “baker” and “baked” can aggregate input and grow activation. However, once each SP has fired once, LISA begins to enforce constraints on activation, requiring structural matching as well as one-to-one mapping, at which time “baker” and “baked” would mutually inhibit one another since “cooker” should not map to two Predicate units in the *recipient*. In the meantime, the lax enforcement of these restrictions continues until LISA finishes running the next SP, “cooked-steak”. To get to that point, after activation has settled (or a maximum number of activation updates has been reached) LISA inhibits all activation to ‘clear the slate’ of each active unit and prevent activation bleed between *driver* SP activations. Once that is completed, LISA again chooses which SP unit belonging to the phase-set gets to be active next. Since in our example, the first phase-set consists of only *cook*(Sam, steak), there are only two SP units, and given the equation in Subsection A.1 which determines the firing order of SPs, and the fact that “cooker-Sam” has recently been active, we know that “cooked-steak” will win out and become active next. Similar to the sequence of activations driven by “cooker-Sam”, *cook*(Sam, steak), “cooked”, and “steak” will all become active. This drives the Semantic units attached to “cooked” and “steak” which in turn provide input to “baked”, “baker”, and “cake” which in turn provide input to their parent SP units and Proposition units.

Now that both SP units have had a chance to be active, LISA will revisit them each twice more for a total of three activations. On these subsequent activations, LISA will begin to preference structural consistency and one-to-one mappings via lateral-inhibition and top-down activation (flowing from Proposition units to SP units to OP units). The second activation of “cooker-Sam” once again activates “cooker” and “Sam” and their respective semantics, however, we will see different activation in the *recipient* analog this time. Once again, input flows into “baker”, “Bob”, and “baked” from the Semantic units, but this time lateral-inhibition between the Predicate units “baker” and “baked” pushes them towards a winner-take-all scenario (preferencing preservation of the one-to-one constraint). At the same time, input flowing from “baker-Bob” into “baker” and from “baked-cake” into “baked” also contributes to their activation. However, “baker-Bob” is receiving input from both “baker” and “Bob” whereas “baked-cake” is only receiving input from “baked”. As both “baker-Bob” and “baked-cake” are SP units, they will each experience lateral-inhibition from one another and fight for activation. Given the greater input to “baker-Bob” which feeds into greater input to “baker” than to “baked-cake” and “baked” respectively, the lateral-inhibition between the SP and Predicate units will settle the network into only “baker”, “Bob”, and “baker-Bob” remaining active. Similarly, when “cooked-steak” once again becomes the firing SP unit in the *driver*, “baked” and “baker” will compete

for activation (as will “baker-Bob” and “baked-cake”) where this time “baked” and “baked-cake” will win, settling out as the active units in the *recipient*. LISA finishes out this phase-set with one more cycle of each *driver* SP unit firing (again with lateral inhibition and top-down activation).

Only once the phase-set is complete does LISA actually calculate mapping. The mechanism by which it accomplishes that is quite similar to ACME’s mapping discovery. LISA has nodes, called Hebb units, which represent potential mappings between any two considered units (one from each analog) of the same type. These Hebb units keep track of which other Hebb units support the mapping they represent as well as which contradict it. For instance, “cooker-Sam” $\leftrightarrow$ “baker-Bob” and “cooker” $\leftrightarrow$ “baker” support each other, but “cooker-Sam” $\leftrightarrow$ “baker-Bob” and “cooker-Sam” $\leftrightarrow$ “baked-cake” are mutually incompatible. While ACME discovers mappings and resolves structural conflicts by settling activation across a network of mapping-units driven by similarity and pragmatic relevance (as defined by the model’s input), LISA’s Hebb units each begin with some input, and receive activation from the most active consistent Hebb unit and inhibition from the most active inconsistent Hebb unit. The input each Hebb begins with is determined by the aggregated mutual co-activation of the units which the mapping the Hebb represents is on.

For example, in the case described above, the Hebb units for “cooker-Sam” $\leftrightarrow$ “baker-Bob” and “cooker-Sam” $\leftrightarrow$ “baked-cake” both aggregated some co-activation since when “cooker-Sam” was first active in the *driver*, both “baker” and “baked” were activated. However, since both “baker” and “Bob” were active, “baker-Bob” would have been more active than “cooker-Sam”, thereby accruing more co-activation. In addition, during both of the subsequent rounds of SP activation, when lateral-inhibition and top-down activation were in effect, only “baker-Bob” would have been active (beyond a momentary blip of activation till the inhibition could kick in), driving an even larger aggregation of co-activation between “baker-Bob” and “cooker-Sam”. As a result, the initial state of the Hebb units’ inputs would strongly preference discovering the mapping “cooker-Sam” $\leftrightarrow$ “baker-Bob” over “cooker-Sam” $\leftrightarrow$ “baked-cake”. However, during this period LISA considers all Hebb units which have had any co-activation during the phase-set, so if there were some additional supporting Hebb units for “cooker-Sam” $\leftrightarrow$ “baked-cake” or inconsistent Hebb units for “cooker-Sam” $\leftrightarrow$ “baker-Bob”, LISA would consider them and pass their respective excitation or inhibition to the Hebb units in question. Furthermore, LISA also considers any potentially already discovered mappings by including the highest weighted consistent or inconsistent Hebb units. These considerations are managed by activating the Hebb units based upon their input, and then using said activation to calculate inhibition or excitation (further changing their inputs) in an iterative fashion till the network settles.

In the example we find that not only are there no strong contra-indicative Hebb units to suggest “cooker-Sam” $\leftrightarrow$ “baked-cake”, we in fact have stronger consistent Hebb units (e.g. “cooker” $\leftrightarrow$ “baker”) for “cooker-

Sam" ↔ "baker-Bob". And since this is the first phase-set, no Hebb units have had their weights set yet (no already discovered mappings). Thus the Hebb unit for "cooker-Sam" ↔ "baker-Bob" would end up active after settling. In a similar fashion LISA's Hebb unit activations would settle with *cook*(Sam, steak) ↔ *bake*(Bob, cake), "cooked-steak" ↔ "baked-cake", "cooker" ↔ "baker", "Sam" ↔ "Bob", "cooked" ↔ "baked", and "steak" ↔ "cake" all becoming active. With these activations, LISA then updates a weight on each Hebb unit such that the weight moves towards the activation value by a model parameter determined learning rate.

Next, LISA would proceed to the next phase-set, activating the only SP belonging to *rest*(steak) three times, once again enforcing constraints after the first. After the phase-set is complete, it would again update mappings in the same fashion, only this time including the already discovered mappings from the first phase-set as given by the weights of their respective Hebb units. In this fashion LISA moves over each input phase-set and discovers mapping locally, while strongly preferencing a global coherence.

Finally, after running on all the Proposition units in the *driver*, LISA calculates a mapping score between it and the *recipient* by ...

### 3.7 Inference

LISA's guiding principles for inference are driven by Copy With Substitution and Generation (CWSG). If some element of a *driver* does not map to any corresponding element in a given *recipient* after LISA has successfully found a mapping between the *driver* and that *recipient* LISA may construct an inferred unit such that the new unit completes the given by being a good candidate for mapping onto the previously un-mapped element. The way that LISA decides when and when not to infer new structures is something which can be set by the modeler. In the general case, if the mapping quality between the *driver* and *recipient* exceeds a set threshold, LISA will then look to generate an inferred unit for any non-mapped element of the *driver*. LISA does this by creating a blank unit of the same type as the non-mapped unit in the *driver*. If the unit is a type which takes semantic connections, it then gets attached to active semantics with weights given by the activity of each semantic unit. Worth noting, this does not do the same thing as copying the semantics of the previously un-mapped unit because other units from other analogs which are active due to connected Hebb Units (i.e. units which the un-mapped unit maps onto in other analogs) can also drive different semantics than the un-mapped unit in the *driver* does. In this way the newly created unit is given a semantic grounding and now must be structurally connected to other units in the *recipient*. LISA centers its connections at the level of SPs, and so when new inferences are made, it is again the SPs which handle the inferred connections. When a new SP is inferred, it will look to find an active Proposition unit running in 'parent-mode' (not as an argument

to another Proposition unit). Since LISA uses temporal synchrony of firing for the purposes of binding units together, the reverse also holds true where if two units are active at the same time, it indicates that they belong bound together – and so our new SP can infer that it should belong to the active Proposition unit. And so, when the SP unit finds an active Proposition unit it is set as the SPs parent Proposition unit since all units of the same type have mutual inhibition and fire in synchrony with their child SP nodes, there should only be one candidate parent unit. The SP then looks for Object, Predicate, and ‘child-mode’ Proposition units<sup>4</sup> which should belong to it. It again only needs to select the active units, since once activation has settled, there should only be one ‘winner’ of each type in the “winner take all” activation method LISA employs. Since LISA doesn’t have any strict enforcement on the “winner take all” aspect of mapping, it may seem too trusting to expect that only one unit of each type will be active. However, it is far more likely than it may at first appear since the uniqueness of mapping is a strong factor in mapping quality which determines whether inference happens in the first place. So, since LISA has decided to perform inference, it is very likely that only one of each unit type will be active at once during the inference step. Once the SP has found its corresponding Proposition, Predicate, and Object (or ‘child-mode’ Proposition) units, the structural connections can be properly inferred and the task of inference is complete.

### 3.8 Schematization

The act of learning in LISA is performed via generalization of multiple analogs into a single new analog which encapsulates the kinds of propositions LISA expects might occur together. This process of producing a schema, or schematization, in LISA is performed in a similar fashion to inference. LISA’s schematization takes place after successful mapping has already happened. For generating a schema, LISA begins with an empty analog and then allows each analog, one at a time, to act as a *driver* and drive inference on the initially blank “schema” analog. Thus, the first analog will drive inference of all its propositions on the blank “schema” analog. Since the “schema” analog is empty, none of the propositions from the *driver* can successfully map onto it, and thus all are marked as candidates for inference. After LISA infers units to map onto all the propositions (and subsequently infers units to map onto their constituent elements: SP, Predicate, and Object Units), then the next analog will take over as *driver*, and drive new mappings to the “schema” analog, potentially forming new inferences. In this way the structure of the “schema” analog is logically similar to the union of all the propositions in the analogs which were used in its construction, which is to say, the “schema” analog helps to answer the question of what is possible to occur in a given situation. In order

---

<sup>4</sup>Proposition units are set to ‘child-mode’ based upon the activation being passed into them from the *driver*, and so we can tell what mode a Proposition unit is in even without it being fully connected to the SP unit we have inferred.



to say what LISA believes is likely to occur, we need the inclusion of a new feature added to LISA, Inductive Confidence which will be covered in Section 4.5.

## 3.9 Comparison to Other Models

Since the bulk of the research presented will either use or be informed by LISA, it is beneficial to explicitly explore in what ways LISA differs from other staple models of analogy. To that end, we will examine the implications of the differences LISA has with SME and ACME.

### 3.9.1 SME

One of the biggest over-arching differences between SME and LISA is their overall complexity. SME’s main functionality is constrained sub-graph matching which is a well studied domain for which many utilities exist. Whereas LISA’s hybrid neural/top-down control architecture can be difficult to understand, communicate, and modify. This is further exacerbated by LISA’s use of neurally plausible architectures where possible which can make certain aspects of the algorithm yet more complex.

To understand the differences between the two models, the most important aspect to study is undoubtedly their mapping algorithms. Since SME and LISA differ greatly on their structural representation, the implementational details of their mapping algorithms are vastly dissimilar. Instead, we should focus on what the differences in capabilities between the two mapping algorithms are:

1. Dissimilar Verbs: SME does not handle mapping between non-identical predicates, whereas LISA uses a semantic encoding to gauge similarity. Instead, SME uses a method which Gentner terms ‘re-encoding’[Gentner, 1989] where a predicate is replaced, when possible, by a more general term as defined in an ontological hierarchy in the hopes that the more general versions of the predicates will then allow a mapping. This disallows what Holyoak would term a far-mapping [Vendetti et al., 2014] where mapping predicates don’t have obvious direct semantic links, e.g. “Sam shoveled the hay with a pitchfork.” -vs- “Bob ate his dinner with a fork.” If those stories had continuations that would be informative to each other, LISA would be able to discover them while SME would fail to find a possible map. However, this freedom in LISA is not without cost since it opens LISA up to finding a wealth of nonsensical mappings based solely upon semantic similarity. The task of deciding what qualifies as a worthwhile mapping is then left up to the continuations of the stories and the map-ability of their embedded contexts.
2. Structural Consistency Enforcement: SME enforces a strict structural consistency between mapped

propositions whereas LISA's mapping algorithm only provides a strong pressure towards structural consistency. This means that in some instances where no structurally consistent mappings exist, LISA can still discover mappings with inconsistent structure whereas SME will instead decide that no mapping is possible.

3. **Different Cardinality Propositions:** SME cannot perform mappings between two propositions with a different number of arguments, which given LISA's division of roles into separate SP Units, LISA handles seamlessly. SME's approach of raising the predicates to a higher level in their ontology does not solve this issue since propositions with a different number of arguments cannot fall within the constrained graph match SME employs. However, in LISA each argument to a proposition is governed by an SP rather than being intrinsic to a Proposition unit, thus the mapping algorithm is agnostic of the difference in argument number and will instead find the best map it can even if arguments are left dangling by the mapping. LISA's behavior here may not always be preferred. However, given the messiness of language we can expect that, in natural text, our propositions will often have disparate numbers of arguments based solely on how verbose the speaker/author is. For this reason, LISA's behavior is likely to yield better results where we want to interpret propositions in known situations with novel syntax. For instance, in the sentence "Sam emptied the dishwasher [of dishes]", we would like to interpret the narrative impact of the sentence as being the same with or without the blocked phrase.

### **3.9.2 ACME**

ACME's difference from LISA is most prominent in its structural makeup. While LISA represents knowledge directly, ACME's representation is instead of potential mappings, the labels of which may be written include knowledge but the actual structure and operation of the model bears no such information. ACME's structure is in fact quite similar to LISA's Hebb Units, which each represent a given mapping and have excitatory and inhibitory connections to structurally consistent and inconsistent mappings' Hebb Units. The implications of this difference are brought to light best by the difference in inference between the two models. Since ACME doesn't have any direct representation of the propositions themselves, in the CWSG paradigm ACME can only perform exact copies of specific elements. This makes schematization across several analogous but disparate analogs which don't share identical elements impossible since schematization requires the model to generalize over analogous elements.

## Chapter 4

# Extensions and Changes to the LISA Model

In the course of research on narrative analogy we found it necessary to revisit the intricacies of the Learning and Inference with Schemas and Analogy (LISA) model and code-base. In the process, it became apparent that a re-writing of the code for the model would be beneficial. In the process of our re-implementation of the code, to improve the model's usability and scope, we added several new features, parameter tuning, and algorithmic modifications. The changes we made encompassed many portions of the model from the usability and features of the interface to the core mapping algorithm. Below we will examine the more substantial changes made to the model and code and their motivation and impact.

### 4.1 Code-base Changes

In re-implementing LISA, we updated the code resulting in several meaningful changes to the utility of the realized model. Initially, when using LISA a modeler would be able to run a simulation of the analogy process while viewing the model's activations on major units in real time. The user could also change the speed at which the model ran to better follow the developments. Unfortunately, direct inspection of the model, or easy modifications or deviations from the typical usage was impossible.

#### **Application Programming Interface**

To assist future modelers, and reduce the barrier to entry for non-expert users of LISA, we moved LISA to a structured callable Application Programming Interface (API) format. This allows users to use the code within whatever wrapper they find most useful for their tasks and datasets. It also makes it approachable for non-expert modelers who want to experiment with different changes to the model (new mapping algorithm, new *driver* activation updating, etc.) since they can treat many of the other methods in the code as black-boxes and drop-in new methods to replace old ones with minimal other changes required.

## Graphical User Interface

Originally, the LISA model used a now deprecated PyGame library, which was both difficult to install on newer machines and, given the lack of support, prone to compatibility issues. To further reduce the barrier to use for new users, we moved the graphics into TkInter, a built-in graphics library in Python.

During this re-implementation, we also developed several new features in the GUI to expand the model’s capabilities. The simplest of these features is a pause button so that the state of activation can be explored statically at any given point during the run. This allows the modeler more control over exploring the emergence of mapping discovery. Along the same line, we added in an inspection feature whereby users can inspect the state and parameters of any major element of the model (semantics are not shown in the GUI, nor can they be inspected directly). This allows for a significantly finer grained analysis of the state of almost every aspect of the model in situ during mapping discovery and inference. The scientific import of these improvements lies in demystifying the causes of unexpected behavior in an otherwise complex system. Before these changes were in place, modelers would have to wonder why unexpected phenomenon in the model arose, and design companion simulations and/or perform deep dives into the code to discover the source of the anomalies.

In addition to the visuals and inspection developments, we also added a more functional, algorithmically impactful, feature to the GUI, namely setting model parameters like activation growth/decay rates, lateral inhibition, or semantic noise. This sounds like a trivial addition, but by incorporating it into the model along with the ability to pause execution mid-run, LISA is now better situated to model cases of cognitive decline during task execution such as through distractions or the introduction of novel stimuli. This opens up the model to exploring new areas of testable hypotheses in the future.

## 4.2 Multi-way Mapping

In the original LISA model, mapping was conceived of, rather naturally, as occurring between two analogs<sup>1</sup>. This is in keeping with the rest of the field; however, it is ill-suited to applications using large scale datasets. To increase the LISA model’s viability as a tool for Artificial Intelligence tasks beyond modeling human cognition, we modified the algorithm to permit LISA to handle mapping onto many analogs simultaneously. This functionality proved indispensable for experiments we conducted on the use of LISA for narrative completion, discussed later in Chapter 6.

---

<sup>1</sup>Except in the case of schema induction where a third empty “Schema Analog” is introduced, see Section 3.8

### 4.2.1 Impact of Multiple Simultaneous Mappings

By considering more analogs simultaneously during the mapping process, our changes impacted LISA's inference and thereby schema induction algorithms. This difference centers on the concept of semantic 'bleed' whereby if there are two or more analogs acting as *recipients*, and not all of them are candidates for inference, the units in a fellow *recipient* can provide semantic activation during the activation of the *driver* which produces mapping.

A concrete example of multiple *recipients* impacting inference can be seen in Table 4.1. When LISA is run on the stories from the table, if the analog about Bill is active as a *driver* while the stories about Joan and Mork are *recipients*, LISA will decide that the mapping between Bill's story and Joan's warrants inference since the analogical match is quite high. Similarly, LISA will discover the natural mapping between the propositions of Bill's and Mork's stories. Thus, when LISA begins the process of inference during activation of Bill's 'Proposition #4' (since there's no corresponding proposition in Joan's story), LISA will also activate Mork's 'Proposition #4' because it has discovered a mapping between them – which leads to a strong Hebb weight connection which spreads activation between analogs. Since both of these will be active, and semantic attachment during inference in LISA is driven by active semantics, the inferred proposition in Joan's story will have a mix of semantics between driving and flying. In the previous LISA model iterations, incorporating multiple analogs would have required multiple separate runs, each with two analogs. However, after the first such run, say between Bill's and Joan's stories, a new Proposition unit would be inferred in Joan's story, and the semantics on the inferred predicate would be solely comprised of those for driving. Then when the next round analogical mapping between Joan's and Mork's stories was run, there would be no un-mapped proposition, and thus no inference by which to incorporate the flying semantics into the inferred predicate. This new behavior at face value is neither a strict improvement nor a strict demerit, since its greatest effect is in softening the primacy bias which the model previously had, which at times helped LISA match with human performance on certain tasks.

### 4.2.2 Changes to Initialization

With the addition of more analogs to the mapping process, we found that LISA became more sensitive to the order of firing of Proposition units for its mapping performance. Specifically, LISA was prone to being trapped by initially optimal mappings which later turned out to be pair-wise incompatible with other, more globally optimal mappings. This issue arose since LISA wasn't considering the mutual incompatibility of the two potential maps during the initial mapping process. More concretely, the units in LISA which govern structural consistency for mapping are Hebb Units, and before the changes to the model, LISA created these

units on the fly when it considered the potential corresponding mapping. However, with the interplay between units not in the *driver*, this became insufficient and Hebb Units needed to be generated for all potential mappings before the model explicitly considered them. This change allowed the model to properly maintain its structural consistency and uniqueness of mapping while computing simultaneous mappings to two or more *recipients*. However, while computationally convenient, we suspect that this is very human-implausible. Requiring that the model consider such a large potential number of joint mappings, and computing their mutual compatibility or incompatibility would strain both working memory capacity and in the cases of large stories, could be quite time-consuming.

### 4.2.3 Results of Multi-Way Mapping

The most important aspect of including the capacity for multi-way mapping in LISA is whether the process works. To gain insight into the performance of our changes to the model, we ran it on several benchmarks with a variety of input analogs. One such example can be seen in Table 4.1. In this run we want to capture the analogical relationship and inferential possibilities between the three stories. The story about ‘Joan’ has a direct predicate match and strong object semantic match to the story about ‘Bill’, and so the analogical alignment there is rather straightforward. However, there is less semantic overlap between the ‘Bill’ and ‘Mork’ stories where ‘Jeep’  $\leftrightarrow$  ‘saucer’, ‘beach’  $\leftrightarrow$  ‘Mindy’s’, and ‘drive\_to’  $\leftrightarrow$  ‘fly\_to’ have meaningfully different semantics. Nevertheless, because LISA’s task is to find the best possible mapping between its input *driver* and *recipient* stories and, while not perfect, there remains positive semantic overlap between the desired object and predicate mappings, LISA correctly discovers the correct mapping and finds it to have a sufficient quality to warrant inference on the ‘Joan’ story. Subsequently, LISA correctly infers from the *driver* story, ‘Bill’, the proposition “drive\_to”(Joan, Civic, LAX), where the semantics of the “drive\_to” predicate are a mix of those from ‘drive\_to’ and ‘fly\_to’. This mix of semantics is due to both of the predicates being active at the time of inference. Since LISA doesn’t make use of selectional restriction, the semantics for the objects cannot restrict the inferred semantics of the predicate and vice-versa.

While this is a promising start, if we are going to use multi-way mapping to handle larger numbers of cases, and potentially deal with real-world datasets, LISA will have to be tolerant of noise in the input, where some stories included for mapping simply do not have a potential analogy with the others. This potential for noise can be mitigated somewhat by carefully, or heuristically, choosing what stories to considering in conjunction when forming mappings.

Table 4.2 is an example of a potentially noisy analogical mapping task where the story about preferring certain products doesn’t really have a good analogy to either of the other two stories. In this case, LISA will

	Joan has a Civic. She wants to go to LAX.	Bill has a Jeep. He wants to go to the beach. He drives his Jeep to the beach.	Mork has a flying saucer. He wants to go to Mindy's. He flies his saucer to Mindy's.
Prop #1	has(Joan, Civic)	has(Bill, Jeep)	has(Mork, saucer)
Prop #2	go_to(Joan, LAX)	go_to(Bill, beach)	go_to(Mork, Mindy's)
Prop #3	want(Joan, Prop2)	want(Bill, Prop2)	want(Mork, Prop2)
Prop #4		drive_to(Bill, Jeep, beach)	fly_to(Mork, saucer, Mindy's)

Table 4.1: A representative Example of multi-way analogy. With the potential for inference onto the smaller Joan analog to complete the story with Joan's desire to go to LAX causing her to perform locomotion (mix of semantics between drive\_to and fly\_to) with the Civic to LAX.

	Joan has a Civic. She wants to go to LAX.	Bill has a Jeep. He wants to go to the beach. He drives his Jeep to the beach.	A person prefers a product made by a company they agree with.
Prop #1	has(Joan, Civic)	has(Bill, Jeep)	makes(company, product)
Prop #2	go_to(Joan, LAX)	go_to(Bill, beach)	agrees_with(person, company)
Prop #3	want(Joan, Prop2)	want(Bill, Prop2)	prefer(person, product)
Prop #4		drive_to(Bill, Jeep, beach)	

Table 4.2: A multi-way analogy example involving noise, where the 'prefer' analog does not match semantically or structurally to the other two stories.

still try to find the best possible mapping that it can, but if the quality of the mapping is below threshold it will not perform inference. When LISA is run over those exact stories it determines that the mappings involving the 'prefer' story are sub-par, and only infers propositions based upon the mapping between the 'Joan' and 'Bill' stories once again producing the correct inference of 'drive\_to(Joan, Civic, LAX)'.

## 4.3 Hebbian Units

One of the most fundamental changes we made to the LISA algorithm is the update to its mapping function. Mapping in LISA is managed through its Hebb Units, which are associated with co-occurring activation of the units which the Hebb Unit codes a mapping for. If two units fire together at the same time, the Hebb Unit connected to the two of them accrues input and eventually grows in weight, helping pass activation between them in the future<sup>2</sup>. To better understand the changes we made to Hebb Units, and why we made them, a deeper dive into the calculations LISA uses for them is needed.

### 4.3.1 Calculating Hebbian Consistency

LISA performs mapping discovery in a roughly similar way to how ACME [Holyoak and Thagard, 1989] does it, and indeed, LISA drew the inspiration for its Hebb Units from ACME. In ACME, the main structural

<sup>2</sup>For how LISA uses Hebb Units in relation to the rest of the model see Subsection 3.6.

units of the model represent potential mappings – much like how a Hebb Unit in LISA represents a potential mapping between two other units in the model. To discover mappings ACME uses links between its units to provide support and inhibition for consistent and inconsistent units. Two units are said to be consistent if the mapping they represent provides structural support to one another, and inconsistent if it violates the joint structural consistency between the two mapping analogs. For example, a mapping between two propositions would get structural support from a mapping between their corresponding arguments, but would be inconsistent with a mapping which took an argument from one proposition to another element not present as an argument in the other proposition. LISA performs a similar task during its mapping calculation stage where Hebb Units excite or inhibit each other based upon the similar consistency rules.

### 4.3.2 Changes to Hebbian Consistency

Previously in LISA’s calculations of inconsistency, two Hebb Units were said to be inconsistent if and only if they represented mappings from the same element to two different elements. This is an intuitive case of inconsistency where the two mappings:  $A \leftrightarrow B$  &  $A \leftrightarrow C$  cannot both be correct since we have to preserve a one-to-one mapping over our elements. However, this is not the only kind of inconsistency we want our model to consider. In order to capture these additional inconsistencies we extended the model, increasing the cases where we label two mappings as being inconsistent with one another to include interactions between mappings at different levels (between different types of units). The logic for these additions is mostly consistent across mapping types; however, there are a few distinctions.

- SP to Prop: A mapping between two SP Units and a mapping between two Prop Units are inconsistent if and only if either one of the SP Units belongs to one of the Prop Units but the other SP Unit doesn’t belong to the other Prop Unit, or if one of the Prop Units is a child of one of the SP Units but the other is not<sup>3</sup>.
- SP to SP: Two SP Unit mappings are inconsistent if and only if either an SP from each mapping belongs to the same Prop Unit, but the other SP Units belong to different Prop Units, or if the two mappings take the same SP unit to two different SP units. To state it more plainly, two role-bindings on the same proposition can’t map to two role-bindings on two different propositions and the same SP unit can’t map to two different SP units.
- SP to OP: A mapping between SP Units and a mapping between Object Predicate Units are inconsistent

---

<sup>3</sup>As a reminder, to represent the recursive structure of language, arguments of propositions can be other propositions, so SPs, which bind argument roles to filler entities, can take propositions as those fillers, called child Prop Units. This recursive capacity needs to be in place to handle things like “Bob knows Sally likes John.” which would be *knows(Bob, likes(Sally, John))*



if and only if one of the OP units participating in the OP mapping belongs to one of the SP Units in the SP mapping, but the other does not. Loosely, two role-filler bindings can't map to each other if their roles or fillers map somewhere else.

The addition of these new inconsistency rules enabled LISA to perform much more cleanly and rapidly over several of its more 'difficult' benchmarks without removing LISA's more human-like errors. For instance, over LISA's benchmark simulation of analogy discovery and analogical inference on stories about a love triangle, the model has historically found proper structural consistency difficult to settle into, especially when lead astray by errant object semantic similarity:

$$\begin{aligned} & \text{loves}(\text{Adam}, \text{Betsy}) - \text{loves}(\text{Betsy}, \text{Chad}) - \text{jealous}(\text{Adam}, \text{Chad}) \\ & \text{loves}(\text{Amy}, \text{Bob}) - \text{loves}(\text{Bob}, \text{Cindy}) \end{aligned}$$

Since the task of finding correct mappings on this type of analogy is highly dependent upon identifying the consistent and inconsistent mappings, using the new Hebbian inconsistency checks, LISA was able to discover the correct mapping substantially faster.

### 4.3.3 Hebb Unit Algorithm

Hebb Units are the representation of a potential mapping between units in LISA. As such, each Hebb Unit is connected to two units of the same type across two different analogs. In LISA, Hebb Units aggregate co-activation during the running of LISA. When it comes time to calculate mapping LISA runs an incremental locally-maximal constraint satisfaction discovery step by settling the activation over mutual inhibitory and excitatory Hebb Units. To see how exactly this is calculated, we first have to look at what exactly the Hebb Units keep track of. Hebb Units have the following main parts:

- **Buffer:** The buffer aggregates co-activation between its attached units by summing over the products of their activation at every given time step during a single phase-set<sup>4</sup>.
- **Input:** The input to the Hebb Unit is calculated at every step of the incremental activation settling process by incorporating information from consistent and inconsistent Hebb Units along with the buffer at the conclusion of a phase-set, and is fed into activation to facilitate weight updating (and thereby mapping discovery).

---

<sup>4</sup>A phase-set is a period of activity in the model before the model tries to discover a mapping. This can be thought of as the collection of propositions which LISA considers "at the same time". Typically this is the model's activation of a single proposition in the *driver*, but it is a parameter that can be set in the model's input firing sequence.

- **Activation:** Activation of the Hebb Unit is calculated during every step of the incremental process right after input is calculated. It is adjusted off of the Hebb Unit's input and current activation, modulated by some model parameters. The Hebb activation is bounded by a hard cap to lie between 0 and 1.
- **Weight:** Finally, after activation over all the Hebb Units settles, all their weights are updated such that the weights move towards the activation by a fixed learning rate set in the model's parameters.

Since the changes we made to this process have to do with the way in which the model updates the Hebb Unit's input, here we will take a more in depth and concrete look at that. Previously the model considered the normalized buffer, activation of the most active supporting consistent Hebb Units, activation of the most active inhibitory inconsistent Hebb Units, and the weight of the heaviest inconsistent Hebb Unit. The equation for this is:

$$\text{Input}_{\text{old}} = \text{Buffer} + \text{ConAct}_{\text{max}} \cdot \mathbf{w}_e - \text{InconAct}_{\text{max}} \cdot \mathbf{w}_i - \text{InconWeight}_{\text{max}} \cdot \mathbf{w}_i \cdot \text{Activation} \quad (4.1)$$

$\text{ConAct}_{\text{max}}$  is the activation of the most active consistent Hebb Unit (Hebb Unit which represents a structurally consistent mapping);  $\text{InconAct}_{\text{max}}$  is the activation of the most active inconsistent Hebb Unit;  $\text{InconWeight}_{\text{max}}$  is the largest weight amongst the inconsistent Hebb Units;  $\mathbf{w}_i$  is a parameter for the models preference to avoid structural inconsistencies; and lastly,  $\mathbf{w}_e$  is a parameter for the preference for structural consistencies.

This is a rather natural and unsurprising way of calculating input with the possible exception of  $\text{InconWeight}_{\text{max}} \cdot \mathbf{w}_i \cdot \text{Activation}$ . This part is included to give preference to already discovered mappings, some of whose units may not have been active in the most recent phase-set, and thus their Hebb Units might have accumulated no buffer, leading to them potentially having no activation despite being meaningful considerations for maintaining structural consistency. The multiplication by Activation is so that while the Hebb is first starting to become active it isn't impeded by previous mappings. Only after becoming active is the input then impacted by other previous inconsistent mappings. LISA does this so that it can explore a breadth of mappings and escape potential local optima and find a better global optimum mapping.

#### 4.3.4 Changes to Hebbian Weight Updating Rule

The changes made to the Input calculation center on the notion that consistent Hebb Units' weights should play a similarly active a role in mapping discovery as compared to inconsistent ones. This rule wasn't previously included since it did not seem to be necessary, or largely impactful to performance since in practice  $\mathbf{w}_i \gg \mathbf{w}_e$ , and the inconsistent weights were sufficient to push the Hebb Units towards the proper mapping.

However, in our use of the model we discovered that, in certain situations, the inhibitory pressure was insufficient to push the model towards the ideal mapping. The occurrence of a structural mismatch from this was heavily dependent upon the order of proposition firing and sizing of the phase sets. However, if the modeler was careless, or unlucky in the case of random firing order, it was possible to trap the model into a state of high weights across two mutually inconsistent mappings which would fight each other but not settle into a structurally consistent pattern. To avoid that, we found that considering consistent mapping weights, and using them to pull the model into a structurally consistent activation made the model significantly less likely to end up in a trapped in a structurally inconsistent local optimum. The change to the equation is rather simple:

$$\text{Input}_{\text{new}} = \text{Input}_{\text{old}} + \text{ConWeight}_{\text{max}} \cdot \mathbf{w}_e \cdot \text{Activation} \quad (4.2)$$

Where  $\text{ConWeight}_{\text{max}}$  is the highest weight of a structurally consistent Hebb Unit.

The input is then used to calculate the change in each Hebb Unit's activation. We found that the process LISA was using to calculate the input led to activation oscillation, and so we introduced a change to stop it and ultimately speed up the convergence of the Hebb Unit settling. The old activation can be seen in Equation 4.3.

$$\Delta\text{Activation}_{\text{old}} = \tau \cdot (\gamma \cdot \text{Input} \cdot (1 - \text{Activation}) - \text{Activation} \cdot \delta) \quad (4.3)$$

Where  $\tau$ ,  $\gamma$ , and  $\delta$  are model constants governing the step-size per iteration, Hebb activation growth rate, and Hebb activation decay rate respectively. Together, Equations 4.2 and 4.3 are used one after another to update the Activation and Input until the process settles, at which point all  $\Delta\text{Activation}$  calculated over all Hebb Units at a given step are below a fixed threshold. The oscillations we discovered were a result of activation reaching 1.

$$\text{Activation} \rightarrow 1 \implies \gamma \cdot \text{Input} \cdot (1 - \text{Activation}) \rightarrow 0 \quad (4.4)$$

Following Equation 4.4, we can see that as activation increases towards 1, Equation 4.3 will yield a negative activation change regardless of how large the input was, guaranteeing that when activation reaches maximum, it will drop at the next time step only to rise once more. To avoid this, we constructed a new activation updating equation as shown in Equation 4.5.

$$\Delta\text{Activation}_{\text{new}} = \tau \cdot (\gamma \cdot \text{Input} \cdot (1 - (\text{Activation} \cdot (1 - \delta))) - \text{Activation} \cdot \delta) \quad (4.5)$$

This seemingly small change of  $(1 - \delta)$  is only referencing a change in the order of operations. In the old equation, we are first incrementing based on the input, growth rates, and how far Activation is from 1 and then

afterwards decrementing by the decay rate and how far Activation is from 0. The new equation is performing the decrement first, and the growth second, allowing the model to actually reach cap and not oscillate as Activation approaches 1.

## 4.4 Changes to Unit Activation

Core to the functionality of LISA is how each unit turns its input into activation. While this fundamental task potentially seems straightforward, LISA employs some non-obvious means involving activation decay and specialized input integrators which aggregate input over larger timescales. In the process of using LISA, we developed improvements for some aspects of the model’s unit activation functions. Most notably, we modified the way in which LISA aggregates input to effect activation on every non-Semantic type of unit in the model as well as the way it computes semantic input to its semantically grounded units.

### 4.4.1 Limits on Activation Aggregation

The way that LISA updates unit activation is by making typically small changes at each time-step. This means that activation in any given unit would build up over time so long as there was some input to it and so long as the growth outweighed the decay<sup>5</sup>. Since LISA yokes decay to activation, there is little decay to offset the slow growth at low levels of activation. When a good mapping exists for LISA to find this poses no problem since the best choice will become the most active and inhibit other competition once lateral-inhibition is turned on. However, when no such ‘good mapping’ exists to find, this behavior ultimately results in LISA reaching a stable state of activation for units which have very little semantic or structural motivation for activation. Unfortunately, since LISA determines mapping fitness by mutual co-activation, middling activation across – potentially several – poor matches can have a deleterious effect on discovered mappings. To combat this, we introduced new functionality to limit the duration of time during which units can aggregate input unless their input reaches a certain threshold before lateral inhibition would be turned on in the model. Specifically, if a unit has not become sufficiently active enough to be recalled into active memory before LISA begins enforcing constraints (top-down activation and lateral inhibition), it can no longer aggregate input and the unit’s activation will decay as in Equation 4.6.

$$\Delta act = -\lambda_{decay} \cdot act \quad (4.6)$$

---

<sup>5</sup>As discussed in Subsection A.2

As a result of these changes, not only can LISA avoid getting caught up in noise, it is also able to be more discerning in its discovered mappings. For example, in the case of the simulation described in Table 4.2, the old activation aggregation resulted in the discovery of poor mappings involving the ‘prefer’ analog whose propositions had low overall semantic or structural support from the other analogs, but would nonetheless slowly build-up activation. However, with the new aggregation technique, no such mappings were found as LISA did not allow units from the unrelated analog to enter active memory in time to sufficiently aggregate co-activation to impact mapping.

#### 4.4.2 Semantic Activation

At conception, LISA was designed with the notion of ontological semantics such as described briefly in Section 3.4. LISA’s semantic units used a Weber sum normalization [Fechner et al., 1966, Marshall, 1995] (a sum utilizing the Weber-Fechner Law which explains the disparity between an actual and a perceived change in a stimulus) to calculate semantic input to its OP units. This worked well on small collections of large-weighted semantic units like those found in hand-coded LISAese, but failed to properly maintain accurate similarity comparisons when individual semantic weights were sufficiently small with respect to the Weber constant, being washed out by the normalization factor. This normalization as implemented in LISA can be seen in Equation 4.7, where the Weber constant is 1.

$$Input(U) = \frac{\sum_{s \in S} s_a \cdot s_w}{1 + \sum_{s \in S} s_w} \quad (4.7)$$

Equation 4.7 is calculating the semantic input to an OP unit,  $U$ , from the set of semantic units,  $S$ . Here  $s_a$  is the activation of the semantic unit  $s$ , and  $s_w$  is the weight from  $U$  to  $s$ . When semantic weights are large, this proves to be a functional semantic activation aggregation method, however, when the weights are small and  $\sum_{s \in S} s_w \lesssim 1$ , the activation function starts to under-represent the semantic input. While most of LISA’s historical human-coded semantic representations use large enough semantic weights for this to not be an issue, it still proves problematic for many word embedding techniques.

#### Cosine Similarity Based Semantic Activation

To allow LISA to make use of small weighted semantic vectors – such as those found in most word embedding techniques – we implemented a different input function which approximates cosine similarity as shown in

Equation 4.8.

$$Input(U) = \frac{\sum_{s \in S} s_a \cdot s_w}{\sqrt{\sum_{s \in S} s_w^2 \cdot \sum_{s \in S} s_a^2}} \quad (4.8)$$

The only difference between this new input equation and using cosine similarity to determine activation is that the set of active semantics are not necessarily identical to the semantic weights of the active unit in the *driver*. As mentioned previously, other already mapped units can impact the active semantics.

## 4.5 Inductive Confidence

A complaint which has been levied against LISA as a model of analogical inference and schematization is that the generated schemas do not sufficiently encompass a “learned” representation of the events[Gentner and Forbus, 2011]. These arguments center around the fact that schemas in LISA place no assumption on the likelihood of any inferred propositions. This is a reasonable complaint. If we are to properly model schematization, the induced schemas cannot simply be a collection of all propositions found in analogically relevant stories, but must have some notion of which propositions are central and which are fringe.

Analogy is good at finding what is possible but poor at deciding what is probable. In order to address this, we have added a new element to the model which we call Inductive Confidence. Inductive Confidence, or IC, acts as a measure of how confident LISA is in the truth value of the knowledge it represents.

As a measure of the truth of a statement, Inductive Confidence only makes sense when applied to Proposition units, not on lower level units since they don’t have truth values (IC can also be calculated on *groups*<sup>6</sup>). For example, in the sentence “John loves Mary.” ‘John’ and ‘love’ as individual units don’t have a truth condition, only when combined into *love(John, Mary)* does a potential truth value present itself.

In order to properly capture the range of beliefs, IC is a value ranging from  $-1$  to  $1$  where  $-1$  corresponds to absolute certainty in falsehood,  $1$  to absolute certainty in truth, and  $0$  to complete agnosticism.

### 4.5.1 Implementation of Inductive Confidence

In general, we want to update inductive confidence such that when we find a good mapping which involves a proposition, our confidence in that proposition increases. Conversely, if we find a good mapping which doesn’t involve a given proposition, our belief in that proposition should fall. However, in the case where no good analogy can be found, we should not update our confidence one way or the other. Inductive confidence must then be updated after considering a potential mapping between stories. So, for a given pair of stories,

---

<sup>6</sup>For a note on *groups*, see Footnote 3 in Section 3.4

to get the resulting IC-changes to each proposition, there are three cases to consider:

1. Case 1: The stories maps well onto each other and the individual proposition maps well onto some other proposition.
2. Case 2: The stories maps well onto each other and the individual proposition does **not** map well onto any other proposition.
3. Case 3: The stories do not map well together in the first place.

Each of these three cases are treated differently. The first case should result in an increase in inductive confidence since a discovered mapping suggests that we have found further evidence that the proposition in question is analogically sound in the context of the story. Conversely, the second case should result in a decrement of some fashion to the IC since we have found evidence of an event not happening given analogically similar context. In the third case, since we do not have an analogous story, we don't want to allow the IC to be updated at all.

To best see this, it may be useful to examine a concrete example. Since Case 3 is trivial and straightforward, we will only speak on Cases 1 & 2. Returning to the earlier analogical mapping example used in Figure 3.1, we are provided with a mapping between "Sam cooked a steak. The steak rested. Sam ate the steak." and "Bob bakes a cake. The cake cools." In the mapping LISA discovers, every unit of the second Analog (Bob's Analog) maps well onto some unit in the first Analog (Sam's Analog), but the reverse is not true as there is no analogous proposition for *eat*(Sam, steak). We will say that the quality of the mappings between each analog to the other (what percentage of units map/how unique the mappings are) is above LISA's model parameter for sufficient mapping to update IC, and so by examining the propositions *cook*(Sam, steak) and *eat*(Sam, steak) we can track the changes to their IC to better understand Cases 1 & 2 respectively.

To calculate the update to the IC of *cook*(Sam, steak) after LISA finds a mapping between it and *bake*(Bob, cake), LISA first calculates the 'evidence' which that mapping provides. That is, how much more credence should LISA place in Sam cooking a steak after considering that Bob baked a cake in the analogous story. This evidence is assigned such that it is based upon the IC of *bake*(Bob, cake) (how confident LISA is that Bob baked a cake in the first place). The evidence is further modulated by how good of an analogy the story about Bob is to the story about Sam as well as how well the individual Proposition units themselves map onto one another. This is done because we don't want to consider a proposition strong evidence for belief in another proposition unless they both map well onto each other *and* their respective analogs make for good analogies. Since the two stories map well onto each other and *cook*(Sam, steak) maps well onto *bake*(Bob, cake), the fact that Bob bakes a cake provides strong evidence for Sam cooking a steak.

LISA then uses this evidence to update the IC of *cook*(Sam, steak) in the direction of the IC of *bake*(Bob, cake). Let us say that *bake*(Bob, cake) has an IC > 0 (LISA has some amount of faith that Bob baked a cake), then it will provide positive evidence for Sam cooking a steak, which we want to use to increase the total IC of *cook*(Sam, steak). LISA tempers this with a learning rate and a growth limiter such that it grows slowly as LISA reaches absolute faith in the truth or falsehood of a statement but is fully receptive when completely agnostic.

To capture these effects we devised several equations which control the growth and decay of inductive confidence in LISA.

$$evidence(P, P^*) = \omega(P, P^*) \cdot IC(P^*) \cdot MQ(A, A^*) \quad (4.9)$$

Equation 4.9 yields the evidence for  $P$  when considering  $P^*$  where  $P$  &  $P^*$  are propositions,  $A$  &  $A^*$  are analogs,  $P \in A$  &  $P^* \in A^*$ ,  $\omega$  gives the mapping weight,  $MQ$  gives the mapping quality, and  $IC(P^*)$  is the inductive confidence of  $P^*$ . Evidence here can be thought of passing the confidence from  $P^*$  to  $P$  based upon how well  $P$  maps onto  $P^*$  and how well their respective analogs map onto each other.

$$\Delta_{IC}(P, P^*) = \eta_{IC} \cdot evidence(P, P^*) \cdot (1 - |IC(P)|) \quad (4.10)$$

Equation 4.10 calculates the change in confidence on  $P$  when considering  $P^*$  where  $\eta_{IC}$  is the IC learning rate. This change depends upon the evidence found from  $P^*$  as well as how confident the model is the truth or falsehood of  $P$ . By including the current confidence of the model as given by  $1 - |IC(P)|$ , we allow the model to disbelieve evidence, especially if the model is especially sure that a given proposition is either true or false. The computational effect of this is that as the confidence reaches its poles (either 1 or -1) the model resist change, however IC is maximally open to evidence when it is close to 0 and thereby unsure of how confident it should be.

Equation 4.10 is only used to update IC when both  $\omega$  and  $MQ$  are above model parameter thresholds. When  $MQ$  falls below its threshold, no updating will occur at all since the stories are not sufficiently analogous, an example of Case 3. However, when  $\omega$  is too small we are instead in Case 2, and see a decay in IC.

To return to our example, we can look at what happens to the IC of *eat*(Sam, steak) which, as it has no analogous proposition, exemplifies Case 2. Since the stories about Bob and Sam are good analogies, *eat*(Sam, steak) will be up for an IC adjustment. However, since there is no analogous proposition in Bob's story, LISA will reduce its confidence that Sam ate a steak. Another way to think of this is that, in the general case, after



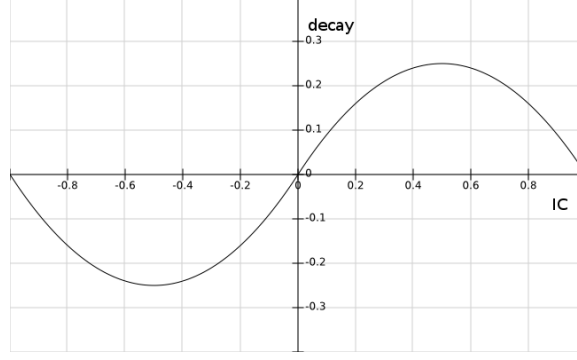


Figure 4.1: The impact to Inductive Confidence decay from current unit confidence.

encountering a story about preparing food where the preparer **didn't** eat the food prepared, LISA now finds it less probable that a preparer will *always* eat the food prepared. This decay to IC is modulated by the same IC learning rate as in Case 1, as well as an additional decay rate and a function of the proposition's current IC such that it decays more slowly near agnosticism and absolute confidence (in either truth or falsehood). To get this kind of behavior, IC decay rate is governed by Equation 4.11

$$\Lambda_{IC}(P) = \eta_{IC} \cdot \lambda_{IC} \cdot IC(P) \cdot (1 - |IC(P)|) \quad (4.11)$$

In Equation 4.11 we describe the process LISA uses to push IC towards zero when it fails to find corresponding analogous structures in an otherwise analogous story. In it  $\lambda_{IC}$  is a fixed decay rate set in the models parameters governing how strict LISA is during this analog intersection discovery process. By using  $IC(P) \cdot (1 - |IC(P)|)$ , we ensure that where LISA is sure about the truth or falsehood, IC experiences little decay, and also as LISA approaches absolute agnosticism it likewise experiences little decay. A benefit of this is that LISA will never cross the zero confidence threshold and start to become confident that a proposition is false simply because it has seen a number of stories without it. Because a different sign of the IC value represents a qualitative change in confidence, a swap can only occur from a good mapping between two propositions with opposite signed IC values. The direct impact on the decay rate as governed by  $IC(P) \cdot (1 - |IC(P)|)$  which, while a kind of parabolic reflection, appears quite sinusoidal in nature as can be seen in Figure 4.1, showing a strong effect closest to  $-0.5$  and  $0.5$  while having little to no decay while near  $1$ ,  $-1$ , and  $0$ . This represents a departure from the sensitivity to change seen in the equations governing Case 1 for which an IC of  $0$  is maximally sensitive.

Using these equations we can update IC on every Proposition unit whose analog was involved in mapping every time LISA changes which analog is the *driver*. We do not calculate IC updates at the conclusion of every phase-set (when mapping happens) since the mapping quality between analogs is still emerging while

the analogs have not yet considered all their propositions. To see this in action, we can again look to some simple example stories.

#### 4.5.2 Tests of Inductive Confidence

For instance, given the stories used in Table 4.2, we can see how IC gets updated as each analog drives mapping.

Story 1:

Bill has a Jeep. He wants to go to the beach. He drives his Jeep to the beach.

Story 2:

A company makes a product. A person agrees with a company. That person prefers that product.

Story 3:

Joan has a Civic. She wants to go to LAX.

If we consider the IC of the propositions in Story 1, we can see all three cases detailed above. The propositions of Story 1 as listed in Table 4.1 are

*P1 : has(Bill, Jeep)*

*P2 : go\_to(Bill, beach)*

*P3 : want(Bill, P2)*

*P4 : drive\_to(Bill, Jeep, beach)*

Since Story 1 and Story 2 form a poor analogy, we'd like to see that no updating of IC happens when considering the two of them. More interestingly, while Story 1 and Story 3 form a good analogy, P4 in Story 1 has no analogous proposition in Story 3. Therefore, we hope that the IC on P1, P2, and P3 would increase after mapping but that P4's IC would decrease since it didn't map onto anything. The actual scores can be found in Table 4.3. While they can be specified, the default initial IC of any proposition in LISA is  $IC = 0.3$ . Given that, after mapping from the "Bill" to "Prefer" stories was completed, all four Proposition units in the "Bill" analog were unchanged as expected. After considering the mapping to the "Joan" analog, LISA updated the IC values, increasing Propositions 1, 2, and 3 but decrementing Proposition 4 again as expected. The same outcome happened when the reverse mappings were considered showing that LISA properly updated its Inductive Confidence to respond to the analogies it found between the three stories. As

	Bill→Prefer	Bill→Joan	Prefer→Bill	Joan→Bill
P1	0.3	0.396	0.396	0.5
P2	0.3	0.399	0.399	0.504
P3	0.3	0.396	0.396	0.5
P4	0.3	0.295	0.295	0.291

Table 4.3: The Inductive Confidence LISA assigned to the Proposition units in the “Bill” story after considering mappings to and from the “Prefer” and “Joan” stories.

can be seen, the actual changes, especially the decrements, are quite small. This is largely the result of the model parameters which are set by default for a slower learning rate and small decay rate with the idea that a single run could have many iterations of consideration to allow the IC to update more fluidly. If the modeler were to desire faster convergence to small and large IC, a small parameter change should afford them that capability.

## 4.6 Conclusion

We successfully expanded the usability and functionality of the LISA model of analogy, opening up the model to greater avenues of research and hopefully allowing it to reach a broader audience than it ever has before. The changes we made to mapping both help LISA find analogies more efficiently with improved consistency checks and also ensure that the model is no longer trapped in deleterious mapping discovery states by incorporating positive bias towards known-good mappings. After all the updates and changes we made to the model’s capabilities, we still needed a means to test the model’s performance on larger scale problems to determine its usefulness to fields outside the scope of cognitive modeling. While we have made in-roads to this by adding in functionality to allow LISA to map between large numbers of stories at once and increasing the model’s flexibility with regard to semantic encodings, we still need a way to fully automate the production of LISAese and move the task of analogy even further away from the hands of the modeler.

## Chapter 5

# Automation of LISA

Analogy is often touted as one of the most central aspects of human cognition [Gentner, 1983, Gick and Holyoak, 1983, Holyoak et al., 1996]. Indeed, it seems that analogy and more generally relational reasoning along with innate linguistic capabilities separate human cognition from that of other primates [Penn et al., 2008]. Understanding and properly modeling this very human process is of importance both to the field of Psychology and Artificial Intelligence. As such, many models of analogy making have been put forward, including SME [Falkenhainer et al., 1989] discussed in Section 2.1, ACME [Holyoak and Thagard, 1989] discussed in Section 2.2, and LISA [Hummel and Holyoak, 1997] discussed in Chapter 3. However, except in very limited domains [Dehghani et al., 2008, Friedman and Forbus, 2009], previous models relied heavily upon human input for encoding narratives from text. This has the obvious drawback that the model’s claims are intrinsically tied to the modeler’s choice of representation. Ideally, these models would automatically generate their own representations from natural text, removing the modeler from consideration and allowing the model to be evaluated on its own merits. In addition, for the purposes of AI, if these models are to be of any direct use on large scale data, their implementation must be automated since hand-coding thousands or hundreds of thousands of stories for every application of the model is prohibitively time-consuming.

Herein we will explore a method we developed and published on to autonomously encode and semantically ground text into LISA’s propositional notation called LISAese [Wilner and Hummel, 2017].

### 5.1 Representational Requirements

LISAese is the format in which analogs and their firing order (how the analogs should be compared with each other to perform mapping) are stored and communicated to LISA. The format includes semantic encodings of both objects and predicate roles (and groups) as well as a predicated structural representation of propositions (and groups). The semantics of objects and predicates consist of a list of named abstract “semantic units” with corresponding weights. For example, in LISAese the sentence “Bob bakes a cake.”, where we have two objects ‘Bob’ and ‘cake’ as well as a two-role predicate ‘bakes’, could have a semantic representation given

by the following:

Bob: human=1.0, adult=0.7, male=1.0, Bob=1.0, baker=1.0, solid\_object=1.0

cake: food=1.0, baked\_good=1.0, sweet=0.9, tasty=0.8, edible=1.0, solid\_object=0.8

bakes: [baker=1.0 decorator=0.6, consoler=0.2] [baked=1.0, eaten=0.9, decorated=0.6]

The semantics presented for the objects are straightforwardly the semantics which LISA will attach to the Object Units it constructs for ‘Bob’ and ‘cake’. The semantics listed for ‘bakes’ are split into two groups, dictated here by the square brackets, to differentiate which semantics should be attached to each role of the predicate which, as noted in Section 3.4, are each represented separately in LISA. Worth noting here is that, unlike during the running of the LISA model, the order of the objects in LISAese propositional notation must therefore matter.

Since LISA treats these semantics as universal, in LISAese, all that matters is the name given to the “semantic unit”, so the ‘solid\_object’ semantic shared between ‘Bob’ and ‘cake’ is in fact the same Semantic Unit in the model. Likewise, the names for each of these units, so long as they are distinct, may be arbitrarily chosen. To wit, the name ‘tasty’ could just as easily have been ‘semantic\_unit\_9’. Since the semantic labels are themselves arbitrary and with the LISA modifications discussed in Section 4.4, it seems a natural step to use distributional semantics or some other word embedding for our semantic representation.

LISAese is sufficient to represent the necessary semantic and structural information LISA needs to make analogies and inferences. With defined semantics on objects and predicates, LISA is able to ground its structural representation, given in a predicated logical format, to represent “Bob bakes a cake.”: *bakes*(Bob, cake). This logical representation is robust to recursive references such as “Jill knows that Bob baked a cake.” which yields: *know*(Jill, *bake*(Bob, cake)). These are written in LISAese to signal what Proposition units LISA needs to create.

## 5.2 Semantic Representation

Ultimately, we need to generate semantic representations for both objects and predicates, but the requirements for automated generation of the two differ substantially. This is because, as previously noted, in LISAese semantics are assigned not to the predicates themselves, but rather to each predicate role independently as was shown above in the LISAese of “bakes” which is split into two verb-roles by the two brackets. This produces a divergence from standard semantic representation which tends to deal with words as singular units. There

is however some previous work dealing with the semantics of verb-roles independently of their verb and each other such as work on selectional restriction. Selectional restriction dictates what kind of objects can fill a given role (or dependency) which, while similar, is not quite the same as what we need for LISA. Instead, LISA needs to know what it “means to be” that verb-role. For example, selectional restriction on the subject of ‘bake’ would dictate that it needs to be animate and a person. However this is not enough for LISA. LISA needs to know what it means to be a baker, which we find to be tied up in either the word ‘baker’ or in what other kinds of actions we might expect a baker to partake in. To put it another way, selectional restriction is concerned with who can do something while what we want to know is what is being done. Conversely, for an object, since LISA does use it as a singular unit in reference to the lexical item, standard word embeddings are quite sufficient.

### 5.2.1 Object Semantics

Over the course of our work on automating LISA, the way we have handled object semantics has undergone several iterations, the greatest shift of which has centered on the change in the LISA model to allow for cosine similarity. This change offered us the opportunity to use dense vector semantics, opening up the possibility for general purpose dense word embeddings and distributional semantics. Prior to this, LISA representations used qualia based semantics, that is, semantic units relating to a word’s formal, constitutive, telic, and agentive roles. These roles can be colloquially understood as “is a”, “has a”, “used for”, and “comes from” respectively. In order to capture either qualia or traditional word embeddings, we turned to outside resources.

#### WordNet

In order to generate some qualia based representations automatically, we made use of WordNet [Miller, 1995], a lexical database with marked hyponym/hypernym and meronym relations. The hypernym relations perfectly capture the “is a”, or formal, qualia role while the meronym covers the “has a”, or constitutive, role. The telic and agentive roles remain outside the scope of this method, however, after studying previous cases of hand-encoded LISA representations, the vastly greater share of semantics are covered by the formal and constitutive roles alone. In order to fill out these relations, and increase coverage of our semantic space (so we get overlap between related objects) we extended these relations with their transitive completions. Hypernymy, and meronymy are transitive with respect to themselves. For example, if we have that an “apple” *is\_a* “fruit” and “fruit” *is\_a* “food”, we can infer through transitivity that “apple” *is\_a* “food”. Similarly, if “car” *has\_a* “mirror” and “mirror” *has\_a* “glass”, we can infer that “car” *has\_a* “glass”. This is not the only

kind of transitivity available to us as meronymy is also transitive with respect to hyponymy (although not hypernymy). Hyponymy is the inverse of hypernymy, or colloquially “such as” as in a “fruit” *such\_as* an “apple”. The transitivity here follows that if the general case has a *has\_a* relation to an object, the specific case must have it as well. Concretely, if “Civic” *is\_a* “car” and “car” *has\_a* “mirror”, then “Civic” *has\_a* “mirror”. To see how hypernymy is not transitive with respect to meronymy, we can look at how an “apple” *is\_a* “fruit” and an “apple” *has\_a* “core”, but it’s not the case that in general all “fruit” *has\_a* “core”.

Using this, we can build up a long list of transitive extensions to help improve semantic overlap between non-identical objects, especially since many seemingly related objects exist at different levels of the WordNet ontology. However, extending this out ad nauseam results in a watering-down of the more central semantics, so we geometrically decrement the contribution of each successive inference by a constant factor, and limit the total considerations to a fixed depth. Based off of experimentation on traditional LISA benchmarks, a depth limit of 5 was found to give sufficient coverage. Using ‘cat’ as an example, a depth of 5 would give us hyponyms of ‘domestic cat’, ‘wild cat’, ‘Manx’, ‘tabby’, ‘cougar’, ‘ocelot’, ‘lynx’, ‘spotted lynx’, ‘common lynx’, etc. While the hypernyms of ‘cat’ to a depth of 5 are ‘feline’, ‘carnivore’, ‘placental mammal’, ‘mammal’, ‘vertebrate’. Using this, ‘cat’ and ‘dog’ meet at ‘carnivore’ at a depth of 2 for each. With geometric down-weighting, by setting the rate at which it scales, you can change how selective the semantics are, but the overall coverage allows most nouns to reach reasonably far up the ontology and most, if not all, of the way down.

While this method provides a viable means of semantic representation, with the addition of a cosine similarity approximation to the LISA model’s mapping algorithm, we can use more robust, distributional, techniques for semantic encoding.

## Word2Vec

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 [Mikolov et al., 2013b] which is produced by a neural network architecture to accurately determine whether two given words are likely to appear within a co-occurrence window in text. 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 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 word. 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 “be”, “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. Using these ‘SemUnit’s, we can produce typical LISAese object semantics from the Word2Vec word embeddings.

## 5.2.2 Predicate Semantics

Similar to Object semantics, the techniques used to encode predicates for LISA have gone through several iterations. However, unlike Object semantics, Predicate Semantics don’t have as many ready-made solutions since the semantics LISA expects are not directly for the verb itself, but rather for each verb-role independently. Capturing the semantics for these verb-roles proves to be a less well studied problem. Since most semantic encoding techniques produce a representation of the verb as a whole, and LISA necessitates distinguishing the semantics on a role-by-role basis, we need to develop our own method.

Rather than base our semantic representation on something like selectional restriction, we want the semantics attached to each verb-role to be about how the object it is attached to interacts with the narrative rather than what type of objects can fill that role. Certainly, selectional restriction on its own would fail to capture most of the important aspects of a predicate, since, for example, ‘cook’ and ‘eat’ have the identical selectional restrictions but are quite different in their impact upon a narrative. However, this alone doesn’t suggest that other, more traditional, techniques are insufficient since one could imagine that an inclusion of some word embedding in concert with selectional restriction would allow for a role-based semantic representation.

Unfortunately, word embeddings based on intrasentential word co-occurrence have some issues when it comes to analysis at the narrative level. For instance, Table 5.1 shows the cosine similarity measures between the Word2Vec embeddings of three predicates: ‘bake’, ‘fry’, and ‘can’. Since the pre-trained GoogleNews Corpus Word2Vec vectors are over lexical entries without any POS tagging, the semantics for “can” will be



	baking	frying	canning
baking	1.0	0.539	0.537
frying	0.539	1.0	0.342
canning	0.537	0.342	1.0

Table 5.1: The cosine similarity scores of the embeddings for “baking”, “frying”, and “canning” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300.

dominated by the modal ‘can’ rather than the verb ‘can’. To circumvent this, the table instead looks at the gerundival forms of the verbs: “baking”, “frying”, and “canning”. As we can see, the difference between “baking” and “frying” -vs- “baking” and “canning” is minimal. This is not that surprising since most things which can be fried or can be canned can also be baked, so the contextual similarities should be roughly equal in measure. However, from a narrative perspective, “baking” and “frying” are more similar than “baking” and “canning” as the stories surrounding “baking” and “frying” are both likely to involve eating and production, whereas “canning” is more likely about storage and preservation. It is for this exact reason that it behooves us to develop actual predicate semantics which are more narrative in focus since our task is inherently narrative in scope.

The first step made into producing a narratively inspired predicate semantic representation came about by using a narrative context window rather than an intrasentential lexical one.

### Co-reference Chains PMI

Since the scope of a predicate’s influence on a given narrative is inherently intersentential, the context window needed to adequately capture meaningful relations would grow too large and be dominated by noise. To avoid this, we need to restrict ourselves to only the most narratively salient aspects of each sentence. Thus, we approach verb-role semantics using narrative chains [Chambers and Jurafsky, 2009]. Narrative chains follow actors through the story, each one comprised of a series of verb-roles filled by that actor as seen in Figure 5.1. The two narrative chains, one following the actor “Bob” and another following the actor “cake” (actor here need not be animate) overlap on the verbs “bake” and “eat”. The set of all overlapping narrative chains when grouped together, form a narrative (any chain which overlaps with any other chain belonging to some narrative also belongs to that narrative).

These narrative chains form a co-occurrence window for verb-roles, and allow us to apply any co-occurrence based statistics to produce a semantic representation of verbs differentiated by role. However, since this technique pre-dates the inclusion of Semantic Role Labeling into our pipeline, the predicate-roles here were approximated by dependency parsing produced from Stanford’s CoreNLP package [Manning et al., 2014, Rudinger and Van Durme, 2014].

Bob baked a cake. The cake cooled. Bob ate the cake.

Bob: nsubj\_bake, nsubj\_eat

Cake: dobj\_bake, nsubjpass\_cool (dobj\_cool), dobj\_eat

Figure 5.1: The above story has two distinct narrative chains, one following “Bob” and the other following “cake”. Because passive voice subjects are narratively similar to direct objects, we encode them as such.

	expire	live
die <sub>W2V</sub>	0.17	0.38
die <sub>PMI</sub>	0.92	0.23

Table 5.2: The cosine similarity scores of the embeddings for “die” vs “expire” and “live” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300 and by Co-reference Chain PMI generated over a corpus of English as a Second Language (ESL) stories using dependencies for verb-roles (bake\_ *nsubj* for ‘baker’) where die<sub>PMI</sub> is for the *nsubj* dependency relation on the verb “die”.

We need to determine a score to measure the statistical independence of any two roles occurring in the same chain. Fortunately, Chambers and Jurafsky, in their initial publication on narrative chains, proposed just such a measure, Point Wise Mutual Information (PMI) [Chambers and Jurafsky, 2008]. PMI is calculated as the log of the joint probability of two events divided by the product of the independent probabilities of those same events as shown in Equation 5.1.

$$PMI(role_1, role_2) = \log \left( \frac{P(role_1, role_2)}{P(role_1) \cdot P(role_2)} \right) \quad (5.1)$$

Calculating the PMI between every pair of roles, we can fill in a PMI matrix where each row (or column since PMI, and therefore the matrix, is symmetric) is an embedding of the verb-role. However, since at the time of this research, LISA was unable to handle small-valued semantics, we restricted each vector to its 50 largest dimensions over the 1000 most frequent verb-roles. The reduction to only the 50 largest values served to make the sparse embedding comprised predominantly of large values (and thus usable by LISA at the time), and the restriction of all vectors to only the 1000 most frequent verb-roles served to improve coverage by ensuring that the vectors, though 95% sparse, were not so sparse as to almost never have overlap.

To motivate this as a reasonable metric, it is worth noting that, unlike most co-occurrence based semantic embeddings, this method is well-built to detect synonym -vs- antonym pairs as shown in Table 5.2. Due to the method’s capture of narrative rather than lexical co-occurrence, the common pitfall that most embeddings run into – namely, that antonyms are used in similar local contexts – is circumvented. However, this feature of the embedding is not central to our interest in it, rather the fact that it directly embeds verb-roles rather than verbs is of significantly greater import as it allows us to use it for LISA.

The dataset used to produce the embeddings was a collection of  $\sim 2500$  ESL stories that were gathered from two online training sites [Lee, 2019, Carlson, 2019] and collated together specifically for this purpose. We used these stories throughout our research, but in all instances they were only used to help train word embeddings rather than as an extension of the training dataset. We used ESL narratives for this purpose for several reasons:

1. ESL stories are not fantastical in nature in that they cover normal day-to-day occurrences. Since the purpose of them is to educate, to be as relatable as possible they often ignore Gricean Maxims and intentionally communicate mundane and otherwise non-novel information which lends itself well to establishing expected trends of real-world scenarios.
2. ESL stories are typically simple in narrative structure. The stories are often written in a linear fashion while using simple straightforward grammar which is easier on the parser.
3. Lastly, while limited in vocabulary, ESL stories don't limit themselves to 'child-like' contrived narratives like many children's stories do. As the intended audience is more often adult, the subject matter is focused more towards realistic scenarios, hopefully improving the usability and transfer of the learned embeddings.

### **Co-reference Chains Word2Vec**

The second iteration of semantic representation for verb-roles was a rather natural extension of the PMI method. In order to improve the robustness of the representations, we explored a typical word embedding technique, Word2Vec [Mikolov et al., 2013b]. We used Word2Vec to produce verb-role embeddings in a 100 dimensional semantic space. Typically, Word2Vec uses intrasentential moving context windows in learning its embeddings. To make use of the narrative structure we explored previously, the context windows we used were over verb-roles in narrative chains rather than lexical entries in a sentence. In practice, this consisted of replacing sentences in our training documents with co-occurrence chains. For example, the story from Figure 5.1 would be represented as two sentences given by each of the chains shown in the Figure. Using these as context windows allows us to capture the same co-occurrences as the PMI method, giving the same improvements over standard embeddings (as shown in Table 5.3), and using Word2Vec comes with some known and tested utility to the produced embeddings. One concern with using Word2Vec in this way is that our training dataset of ESL stories, while initially not especially large at roughly 2500 short stories, becomes substantially smaller when we restrict it to only its verb-roles. However, on small scale benchmark LISA simulations, these embeddings proved to be sufficiently usable to induce desired mapping.

	donate	take
give <sub>w2v</sub>	0.34	0.59
give_A0 <sub>roles</sub>	0.15	0.09

Table 5.3: The cosine similarity scores of the embeddings for “give” vs “donate” and “take” as generated by Word2Vec on the GoogleNews corpus with a dimensionality of 300 and by Word2Vec embeddings using co-reference chains generated over a corpus of English as a Second Language (ESL) stories using SRL for verb-roles (give\_A0 for ‘giver’). Here we see the same effect as with PMI in Table 5.2. We use different verbs show that the same property holds with transitive verbs as well.

As this method was employed before the changes to LISA’s semantic activation function allowing for small dense semantic values, we needed to establish a means of sparsifying and increasing the resulting vectors. To that end, we turned to the work of Faruqi et. al in producing sparse overcomplete vectors from Word2Vec embeddings which improved performance on several semantically driven tasks [Faruqi et al., 2015]. When implemented on our verb-role embeddings, we increased the dimensionality by 10 fold to 1000 and found the resulting vectors to be 93% sparse, putting them roughly in line with the sparsity of the PMI method. With this method, we were on somewhat more stable theoretical ground, however, our coverage was significantly reduced due to the limitations of the training set. More qualitatively, these embeddings have the shortcoming of being obtuse and not human readable, which while not an issue for their use, becomes difficult to diagnose in their error. As now, when presented with a semantically driven failure, it is unclear if the issue rests at the level of the theory or only in the practice of training sparsity for one of the constituent roles involved in the error.

### Propbank Word2Vec

The final iteration of verb-role semantic encoding was conceived of as an attempt to bridge some of the gap between the two previous methods. The goal was to introduce some interpretability via a hand-coded knowledge base along with the coverage and power afforded by Word2Vec. In order to accomplish this we turned to PropBank and FrameNet [Kingsbury and Palmer, 2002, Baker et al., 1998] which are hand compiled resources containing, amongst other verb information, predicate role descriptions for most common predicates. To encode verb-roles then, all we need to do is encode their given role descriptions. Since these descriptions are natural text (e.g. “baker”, “creation”, “ingredients”, and “benefactive” for the roles on the verb ‘bake’) we can turn to traditional embedding techniques such as Word2Vec once more. Also, since the descriptions are written in general English, we can make use of the Google News Corpus pre-trained Word2Vec embeddings into a 300 dimension semantic space. As above, since these embeddings were dense, and at the time LISA required sparse semantics, we used the same sparsification technique as for co-reference chains with the same 10 fold increase to dimensionality giving us a 3000 dimension semantic space.

This encoding paradigm helps to address our previous concerns since role descriptions in FrameNet are directly human-interpretable and salient to the role in question. For example, in the first sentence of our example story “Bob baked a cake.” ‘baked’ has two roles filled by ‘Bob’ and ‘cake’. The semantic role ‘bake\_A0’ (filled by ‘Bob’) is described in the “bake” frame of FrameNet as “baker” and the role ‘bake\_A1’ (filled by ‘cake’) is described as “creation”. So, to get an embedding of what ‘bake\_A0’ means, we can instead turn to what “baker” means and use traditional word embedding methods. To avoid getting LISA’s semantics jumbled up between predicates and objects, which can be a concern during its inferential steps, we separate the two semantic spaces entirely by treating them as entirely different dimensions.

Since PropBank frames are separated by role, this technique now forces our model to rely upon semantic role labeling rather than dependency parsing. In order to capture the associated roles and their fillers, we employed the Illinois Semantic Role Labeler which is the Semantic Role Labeling (SRL) part of the UIUC Cognitive Computational Group’s tool suite [Punyakanok et al., 2008]. Of potential concern is that the frames’ role descriptions are further separated by verb-sense which introduces the problem of Word Sense Disambiguation (WSD) to the task. To this end, we looked into various feature sets for Verb Sense Disambiguation (VSD), a sub-discipline of Word Sense Disambiguation (WSD), including syntactic and semantic features such as local dependency tree structure, negation, Named Entity Resolution (NER) of dependent nouns, and WordNet [Miller, 1995] super-sense of dependent nouns. We also explored the use of narrative chains as a discriminating metric, however all such models proved to be ill-suited and were substantially outperformed by the included VSD of the Illinois Semantic Role Labeler. And so, we instead made use of the suggested verb-roles directly from the SRL engine itself. Together, these tools and libraries allowed us to obtain role descriptions ready for Word2Vec embedding into a predicate-semantic vector-space, replacing the narrative structure we got from narrative chains with the narrative information contained in the knowledge base of PropBank.

## 5.3 Structural Representation

With semantics handled, we still need to ensure that LISA properly attaches all the symbolic units corresponding to the objects and predicate roles. To do this, we must reduce the narrative to a sequence of propositions over those same objects and predicates. This is broken apart into three main parts: identifying predicates and their arguments, figuring out which arguments are co-referential, and finally bringing it all together to construct the LISAese for the narrative.

### 5.3.1 Predicate Role Filling

In order to encode language in the kind of predicated logic LISAese requires, we need to identify predicate roles and the objects that fill them. In general, identifying predicates is rather straightforward with simple shallow parsing as we need only identify the verbs (we don't deal with nominalization). However, properly attaching objects to roles relies upon a more complex, deeper parsing. In general, we experimented with two main avenues for role filling: dependency parsing and semantic role labeling (SRL).

#### Dependency Parsing

Dependency parsing is an analysis of the grammatical relations between syntactical elements of a sentence. For example, in the simplest case, we may be presented with text such as “Bob loves Jill”. A dependency parser would recognize that ‘loves’ is the root of the sentence, and has dependent relations to ‘Bob’ and ‘Jill’ as its subject and direct object respectively. To capture this, we used the Stanford CoreNLP package [Manning et al., 2014], making specific use of their lemmatizer and obviously their dependency parser. Lemmatization is the task of removing inflections, pluralizations, and other declensions to return the base form of any given word. For example, the lemmatized form of our previous sentence would be “Bob love Jill” since the base form, or lemma, of ‘loves’ is ‘love’. Using both of these tools allows us to determine the two lemmatized grammatical relations: ‘love’  $\xrightarrow{\textit{nsubj}}$  ‘Bob’ and ‘love’  $\xrightarrow{\textit{dobj}}$  ‘Jill’. We can then use these to form LISAese propositional notation as will be describe in Subsection 5.4.

#### Semantic Role Labeling

There remain two substantial issues with our use of dependency parsing for attaching objects to verb-roles:

1. Confounding of verb-role attachment:

We should note that grammatical dependencies are insufficient to universally determine predicate attachment. While the *nsubj* role of a predicate might often correspond to the agent effecting an action, not only is that not always the case, but even within a single verb (and verb sense) the same dependency relation can have substantially different semantic relations. A simple example of this can be found in the following two sentences:

“Sam broke the glass.” -vs- “The glass broke.”

Dependency relations for *nsubj* on both of these sentences are ‘break’  $\xrightarrow{\textit{nsubj}}$  ‘Sam’ and ‘break’  $\xrightarrow{\textit{nsubj}}$  ‘glass’ for the first and second sentences respectively. This is a clear issue since in every meaningful way, the “glass” in the second sentence is like the “glass” of the first, not like “Sam”. If all our system had to go off of were

dependency parses, there is no way it would be able to properly intuit that the *nsubj* relation in one case should be confounded with the *dobj* relation in another. It may at first appear that all we would need to do would be to keep track of what dependencies are present on a given verb, however this too is insufficient as we can see if we instead look at the two sentences:

“Sam ran the race.” -vs- “Sam ran.”

Here we have the same change in what dependencies are present on the verb as in the previous example, but this time it is without the same shift in role attachment. If we wanted to push dependencies further we could solve this case, and indeed most like it, by also incorporating the transitivity of the verb form, but to do this we would need to capture verb sense as well as maintain a list of rules for when and how to shift dependencies on a verb-by-verb basis. The more post-processing we perform, the less what we are using looks like standard dependency parsing. Moreover, no amount of verb-based dependency shifting rules would help account for role changes that are independent of verb form as is the case in our second issue.

## 2. Imprecision of dependency relations:

Dependency relations are too imprecise to properly capture the kind of relations we need. Basic dependency relations only allow us to capture a vague prepositional relationship, although enhanced dependencies keep track of the preposition used, this alone is not enough. For example, we can see different semantic relations marked by the same preposition in the two sentences:

“Sam crashed his car before the exit” -vs- “Sam crashed his car before noon”.

In the first, the prepositional phrase “before the exit” governs a locational relation to the predicate “crash”; whereas, in the second “before noon” is temporal in nature. The dependency relations in both sentences are both the same nominal-modifications as shown by ‘crash’  $\xrightarrow{\text{nmod}}$  ‘exit’ and ‘crash’  $\xrightarrow{\text{nmod}}$  ‘noon’. Even if we were to use enhanced dependency relations, since the preposition in question for both sentences is the same ‘before’, there remains no discernible difference. As such, the dependency parses for both sentences are indistinguishable and thus too imprecise.

To handle these two issues, we turned to a different parsing technique, Semantic Role Labeling (SRL). Sometimes referred to as shallow semantic parsing, SRL’s goal is, for each predicate in a sentence, to label phrases with their semantic relation (e.g. ‘goal’, ‘location’, or ‘recipient’) with respect to that predicate. While the specificity of the semantic roles labeled differs from implementation to implementation, each SRL engine captures our desired precision while allowing for distinction between otherwise confounded object to predicate role attachment. To wit, using SRL on each of our example sentence pairs yields both the same

semantic role for “glass” in both sentences in our first example and different semantic roles for “exit” and “noon”.

As for actually implementing Semantic Role Labeling for our task, we used the SRL engine in Cognitive Computation Group’s (CCG) Illinois Parser [Punyakanok et al., 2008]. The labeler makes use of joint inference over several SRL systems with Integer Linear Programming for global constraint satisfaction to optimize its labels. The labels generated are over spans of text and belong to one of four categories: verbs, a fixed argument list (i.e. *Arg0*, *Arg1*, etc.), modifiers (i.e. adverbial, negation, location, temporal, etc.), and referents/extensions such as the “who” in “Let he *who* is without sin cast the first stone”. Of these, most are fairly self-explanatory with the exception of the fixed arguments. The argument labels themselves are not human readable, but rather tie into the fixed semantic roles for a given verb. For instance, in the referent/extension example sentence, SRL parsing would yield the relation “cast”  $\xrightarrow{\text{Arg1}}$  “stone”. When taken alone, “Arg1” doesn’t provide much in the way of human meaning. However, with the additional knowledge that the “Arg1” role for the verb “cast” is always the “thing thrown”<sup>1</sup>, it paints a much more compelling picture.

For a concrete example of the kinds of labels SRL can produce, we will again turn to the example sentence “Bob loves Jill”. As discussed above, we have a single predicate with two objects which need to be properly attached to verb-roles. Unlike above however, the roles in question will be semantic rather than grammatical in nature. The resulting parse returned by the CCG SRL engine is “loves”  $\xrightarrow{\text{Arg0}}$  “Bob” and “loves”  $\xrightarrow{\text{Arg1}}$  “Jill”. Again, with the knowledge that the *Arg0* and *Arg1* roles of “love” are always the lover and the one loved respectively, this allows us to have a representation much closer to the actual meaning than simple grammatical relations would provide for.

The SRL usage described so far accounts for simple sentence structure rather well. However, there still remain two issues we need to tackle for a more complete representation, namely recursive propositions and copular verbs. Recursive propositions, as discussed previously in Section 5.1, are needed to cover sentences such as “Jill knows that Bob baked a cake”. Since the predicate ‘know’ here has two verb-roles, but only one of them is filled by an object, the methods we’ve discussed so far will prove insufficient. In order to fill the *Arg1* role, or “thing known”, we need a way to properly attach the predicate ‘bake’. In order to accomplish this, we pair our SRL label with the closest matching dependency parse component to identify if it is either a noun phrase or a verb phrase and attach the LISA unit corresponding to the head word of that phrase, either an Object Unit for a noun or a Proposition unit for a verb. As another example, we can see the same structure in the sentence “Jill knows Bob loves her”. Here the structure we hope to capture is that “Jill” is the agent and *love* is the patient of *know*. SRL helps us discover these attachments by labelling “Jill” as the *Arg0* and

---

<sup>1</sup>This is the actual role description from PropBank’s corresponding frame.



“Bob loves her” as the *Arg1* spans of text for the predicate *know*. Similarly, “Bob” is identified as *Arg0* and “her” (co-referentially “Jill”) as *Arg1* of *love*. Using these labels, we check to see if the range of text of the parse constituents corresponding to the proposition *love(Bob, Jill)* is within the span of text labeled as *Arg1* of *know*. Since it is the same, we would then attach the proposition itself as the argument.

### 5.3.2 Narrative Chains

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 align our Proposition units with Object Units such that we associate all mentions of the same object to a single Object Units. This includes two pieces: anaphora resolution and co-reference resolution.

Anaphora resolution, sometimes called pronoun resolution, consists of discovering the associated entity a pronoun refers to. For instance in the sentence “Jill hates coffee. It’s too bitter for her taste.” Anaphora resolution would identify that “her” refers to “Jill”, allowing for a proper alignment and association of propositions and adjective modifiers to their correct Object Units.

Typically a more difficult task than Anaphora resolution, co-reference resolution governs aligning different phrases used to mean the same entity. For example, in the two sentences “IBM is one of the oldest machine focused companies in the US. The company was founded in 1911.” Co-reference resolution would yield “IBM”  $\xrightarrow{\text{co-ref}}$  “The company”. Similarly to anaphora resolution, this allows us to properly associate objects across a narrative.

To get anaphora and co-reference resolutions over our documents we looked again to Stanford’s CoreNLP tool suite [Manning et al., 2014] which has both of these baked into their NLP pipeline. Once we have the chains of entity references resolution produces along with their associated verb-roles (either SRL or dependency parse based) we have constructed what Chambers call narrative chains [Chambers and Jurafsky, 2008, Chambers and Jurafsky, 2009]. These are chains of predicate-roles which are filled by a single entity. By combining chains which have at least one overlapping predicate, we can construct cohesive narratives. For the purposes of generating LISAese, we will instead assume that what should or shouldn’t belong to the same narrative will be dictated by the modeler; and so, all propositions given to the system that the modeler says go into one analog will belong to the same single analog.

## 5.4 Constructing LISAese

The last step in the whole process is actually producing the LISAese coding to allow LISA to implement the identified propositions. The LISAese representation covers multiple analogs as well as how LISA should run them, dictating what order analogs should drive analogy discovery, what order each proposition should fire within each driving analog, when LISA should stop to consider mappings, and lastly when LISA should attempt inference.

For each analog that we want LISA to consider in its analogy discovery, the LISAese needs to cover the definition of Object, Predicate, and Proposition units. Object Units are defined by both an identifying name, as well as a list of Semantic unit names with associated weights. These semantic units and weights may be produced by either of the methods discussed in Subsection 5.2.1, however we have found the best performance with Word2Vec representations. Similarly, Predicate Units are defined by an identifying name as well as a list of semantics given in one of two ways. Either the semantics are given in a role-by-role fashion where each role receives distinct, possibly overlapping, semantics, or the semantics are what LISA calls ‘auto-defined’ wherein a single list of semantics and associated weights is provided along with a number of roles and LISA then constructs independent role semantics by appending a role-label to the end of each semantic identifier, creating distinct Semantic units for each role. While it is simpler and permits standard verb embedding techniques, auto-defining predicates in this way does limit the utility of the representation as cross-mapping, or mapping across different roles, can no longer be semantically motivated. For instance, given the two predicates ‘chase’ and ‘flee’, it is obvious that a cross-mapping of roles is sensible in that the roles of ‘chaser’ and ‘chased’ are similar to the roles of ‘one fled from’ and ‘one who flees’ respectively. And so, to properly capture any role-inverted semantic relations, we need to have role-by-role semantic encoding, for which we can turn to any of the methods discussed in Subsection 5.2.2. In practice, on hand curated data we have found success with all of them, but the best performance on these benchmark simulations was with the PropBank-Word2Vec-technique. Lastly, for defining Propositions units, for reasons discussed above and to pair with the PropBank Word2Vec technique in use for Predicate Units, we use Semantic Role Labeling to manage structural representation.

With all the analogs defined, all that remains is telling LISA the sequence in which to fire driving units. To do this, we default to a simple method where inference is turned off and every analog gets to be the *driver* once with every other analog as its *recipient*. Within each driving analog every proposition fires once with mapping weights updated after each firing. This firing order is not sufficient for all types of analogy discovery, nor all model instances, and different firing schemes are still left up to the modeler to determine.

With sequence definition completed, we have a sufficient method 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 correctly function on the resulting LISAese. In order to test this, we will explore three benchmark simulations LISA has successfully completed in the past: Latin to Logic, Love Triangle, and Driving a Vehicle.

## 5.5 Simulations

To see how our automated production of LISAese works, we will dive straight into examining what the benchmark simulations are and LISA’s performance on the automated versions of them. In our exploration, we will only present the propositional notation for the encodings since the actual semantics are prohibitively large. For the first benchmark, “Latin to Logic” [Hummel et al., 2014], the original proposition 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)`

The proper analogy, which LISA discovers, maps between Second Order Logic and Natural Language and infers that Second Order Logic has unprovable statements and that it is incomplete.

In order to test our automatic representation, there is one more step of modeler input we need. To be able to run our automated encoding, we must first identify English sentences which convey a similar meaning to the propositions in the benchmark. 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:

`is_incomplete(Second_Order_Logic)`

Analog 2:

`is_able(Latin, to_self_reference)`

`prove(Latin, a_statement)`

disprove(Latin, a\_statement)

is\_incomplete(Latin)

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<sup>2</sup>. Worth noting is that the English description we chose gave us different propositions from our original representation. Nonetheless, it conveyed sufficiently similar meanings and the correct mapping and inference was still produced.

For “Love Triangle” [Hummel and Holyoak, 2003] the hand-coded benchmark structure and the automatic structural encoding were identical, as shown below:

Analog 1:

*love*(Sally, Jack)

*love*(Jack, Megan)

*hate*(Sally, Megan)

Analog 2:

*love*(John, Mary)

*love*(Mary, Bob)

Again, here we needed to select sentences to generate the encodings from. We chose our sentences rather straightforwardly with Analog 1 produced 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 to inform proper mapping. In “Love Triangle”, while the structure remains simplistic, the object semantics are inversely correlated with the ideal mapping and the correct mapping is only discoverable by enforcing a one-to-one mapping across two propositions. Specifically, the 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 addition, the ideal mapping is not discoverable when each proposition is viewed in isolation since for example *love*(*Sally, Jack*) aligns perfectly well with *love*(*Mary, Bob*) despite not being the correct mapping. To deal with this, LISA must be able to consider both the *love* propositions together before determining mapping. This is an instance where LISA’s success or failure for the problem still lays at the hands of the modeler, which we argue it should since controlling the method and means by which information is available to influence mapping is a strength of the model. To ensure that LISA does solve this mapping, the sequence section of the LISAese representation needs to

---

<sup>2</sup>Of course, proving and disproving the same statement would make a logic inconsistent, not incomplete, but for LISA’s purposes only the analogy and its inferences matter.

include joint firing of both *love* propositions before mapping weight updating. With the proper sequence, in both the hand-coded version and the auto-encoded version, LISA finds the ideal mapping and produces the correct inference, namely *hates*(John, Bob) in Analog 2.

Lastly, we will look at the most complicated benchmark, “Driving a Vehicle” [Hummel and Holyoak, 2003] which incorporates propositions taking other propositions as their arguments. Its hand coded structure is shown below:

Analog 1:

```
has(Joan, car)
want(Joan, goto(Joan, airport))
```

Analog 2:

```
has(Sam, Jeep)
want(Sam, goto(Sam, beach))
drive(Sam, Jeep, beach)
```

We once again turned to rather simple sentences to capture the stories: “Joan has a car. She wants to go to the airport.” and “Sam has a Jeep. He wants to go the beach. Sam drives his Jeep to the beach.” The automatic encoding produced a very similar structure for these two stories as the original hand-coded benchmarks:

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)
```

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”. When inference was turned on in the model, LISA also managed to produce the correct proposition – namely, that Joan drives her car to the airport.

## 5.6 Concluding Thoughts and Model Limitations

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 this incorporation of a linguistic front-end, LISA is now the first computational model of analogy capable of producing semantically-grounded

encodings of 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 and failures on modeling tasks reflect its fundamental tenets, rather than the representational and modeling choices made by its modeler.

Despite these successes, there is still a great deal of room for improvement on the linguistic front-end. Most fundamentally, the proper encoding of a story relies upon the simultaneous success of all the NLP tools in use. If the dependency parser, lemmatization, named entity resolution, co-reference resolution, semantic role labeling, or verb-sense disambiguation have an error, the proposition, predicate, or object which they mislabel will be improperly encoded into LISAese. While all the models our system uses have relatively high fidelity themselves, the odds that **none** of them have an error for a given story is significantly lower. This results in a substantially lower performance of our encoding method for grammatically complex stories.

Another limitation of the model is that the techniques described above aren't really sufficient to handle capturing negation. While the SRL and dependency parsing labels do capture negation modifiers and their attachment (either to an object or a proposition), there still remains the question of how to structurally and semantically represent that negation in LISA. At the moment, negation is treated as a semantic modification for Object Units and a modifier role on predicates, but this is fundamentally flawed. To see how this fails to capture the proper structure needed for analogy, we can study the following sentences where seemingly similar constructs would suggest different structural representations to discover proper analogical mappings.

“How I’m feeling is not angry.” -vs- “I’m not angry.”

While the first sentence is somewhat awkward, the purpose is to show that two different negated adjectives, both acting as copular subjects (albeit clausal vs nominative respectively) can have meaningfully different interpretations for analogical purposes. This difference is centered on the implicature behind the phrasing. In the first sentence, the implication is that there exists some charged state that the speaker is in, which is substantially different from anger. Whereas the second sentence purely suggests that the speaker is not angry and perhaps has no emotionally charged state whatsoever. To capture these differences requires a better understanding of contextual usage and the potential expected emotional states of the speaker.

Lastly, the encoding system has no meaningful temporal ordering to the encoded events. LISA itself doesn't natively have any explicit encoding for time, however, in future experiments, chained group structures could potentially be used to capture something similar to a partially or locally ordered temporal relation. As it stands now, this limits some of the inferential capabilities of the model over the automatically produced encodings; and while the weakness is also present in normal hand-coded LISAese it remains a weakness of the system at large nonetheless.

## Chapter 6

# Story Cloze

Natural Language Processing has proven indispensably useful in recent decades with the incorporation of question answering, web search, and automated NLP assistants into our everyday lives. As its capabilities have increased, so too have its applications. To date, progress in NLP has largely been limited by the unapproachability of intersentential discourse and the external knowledge needed for pragmatics. In order to better utilize NLP’s potential, researchers would like to solve tasks aimed at these higher order domains; however, these tasks remain intractable to models despite a wealth of data. We propose that this difficulty is, in large part, related to the nature of the added complexity and lack of generalizability inherent in cross-sentential constructs. In this chapter, we present a method for approaching these high level tasks by using a hybrid of instance-based and aggregate-based learning to overcome issues of generalization and coverage over our dataset. In order to make use of both individual instances and generalized knowledge structures, we will present a model utilizing narrative analogy to compare test cases to specific training examples and cluster them to produce schema-like proxies.

By better understanding the uses, strengths, and pitfalls of narrative analogy we can help direct the further research and application of analogical features towards more fruitful ends. Here we will look into the application of narrative analogical features on a popular task, the Story Cloze Task, and analyze if/how it is most useful as well as attempt to identify what further work is needed to solidify the utility of analogical measurement for story-based tasks at large.

### 6.1 Story Cloze Task

In order to measure our model’s ability to capture higher order properties, we need a task from the literature which emphasizes discourse, preferably through narrative structure. Toward that end, we look at the Story Cloze Task put forward by Mostafazadeh et al. [Mostafazadeh et al., 2016a]. The Story Cloze Task is a discernment task where the model is asked to choose between two alternate endings to a story. The stories are all realistic and composed of five sentences which are sensibly connected with nothing extraneous or

unrelated included. The stories are presented to the model without the 5<sup>th</sup> sentence and our task is to choose the correct last sentence versus an implausible but locally coherent alternative. The following example from the dataset illustrates one such presentation:

There was an extremely obnoxious boy in Kerry’s economics class. He monopolized every class discussion with his mediocre ideas. The professor could never get him to be quiet. One day, he showed up to class with laryngitis.

For this story, the last two sentence options are: “Kerry was disappointed.” –or– “Kerry was so grateful!”. The obvious correct completion is the second sentence. Nonetheless, coming to that conclusion is quite difficult using only local features. In general, the implausible alternative sentences are supposed to be constructed such that word choice, standard word embeddings, and sentiment are counterbalanced and therefore poor indicators for discernment. In practice, author biases ended up introducing a range of low-level features which were highly salient to the task.

### 6.1.1 Origins of the Task

The Story Cloze Task is the successor of the Narrative Cloze Task which was put forward by Chambers and Jurafsky [Chambers and Jurafsky, 2008]. Narrative Cloze involves presenting a story with an event removed and testing a model on its ability to suggest replacements. The task was presented as means to objectively measure schema or scene comprehension systems. As such, the Narrative Cloze Task itself was largely unimportant since it had no real world applications, nor were there any which used it as a component. Instead, the interest it generated was for the ability to measure scene comprehension in a rigorous and approachable format. Unfortunately, it was found that deeper scene learning methods could be outperformed by lower level language modeling techniques such as in Pichotta & Mooney [Pichotta and Mooney, 2014] and Rudinger et al. [Rudinger and Van Durme, 2014]. Story Cloze was presented in the hopes that it would foil simpler approaches and require a richer understanding of scenes since similar language modeling techniques the original authors tried failed to produce results significantly above baseline. The most notable baselines presented in Mostafazadeh et al. are presented here:

1. Word2Vec [Mikolov et al., 2013b] in which word-by-word embeddings are averaged to produce a vector representation of the first 4 sentences as well as the candidate 5<sup>th</sup> sentences
2. Skip-thoughts Model [Kiros et al., 2015] that uses a Sentence2Vec embedding averaged over the first 4 sentences compared to each embedding for the candidate 5<sup>th</sup> sentences



Model	Accuracy
Chance	0.5
Choose 1 <sup>st</sup>	0.513
Word2Vec	0.539
Skip-thoughts	0.552
DSSM	0.585
Human	1.0

Table 6.1: Performance on Story Cloze Task. Choose 1<sup>st</sup> is shown to reflect that, in the data, the first listed possible 5<sup>th</sup> sentence is slightly more likely to be correct.

3. Deep Structured Semantic Model [Huang et al., 2013] that uses two deep neural networks. One network embeds the first four sentences into a semantic space. The second embeds the candidate final sentences into the same space

All models produce candidate vector pairs for the two possible 5<sup>th</sup> sentences which, via cosine similarity to a context vector representing the first 4 sentences, are used to discern the most likely story completion. In the end, the models demonstrated a greater than chance performance, but in general were rather poor at solving the task when compared to human gold standard as can be seen in Table 6.1.

More recently, the Story Cloze Task garnered a significant amount of attention as it was the shared task for LSDSem2017 [Mostafazadeh et al., 2017]. Prior to LSDSem2017, most of the interest and publications regarding the Story Cloze dataset focused mainly on the data itself rather than the task. This interest includes publications using the causal and temporal relationship between sentences for context inference [Zhang et al., 2016], or labeling the data with temporal event relations [Mostafazadeh et al., 2016b]. However, since the announcement of its inclusion in LSDSem2017, there have been a great many papers focused on solving the Story Cloze Task.

### 6.1.2 Previous Approaches to Story Cloze

While there were some very promising approaches to the Story Cloze task which embraced the spirit of the task by focusing on narrative comprehension [Chaturvedi et al., 2017], there were also several ‘simpler’ language modeling based solutions which performed as well as or better than the knowledge heavy solutions. In fact, many of the leading solutions, even if they used narrative features, heavily incorporated these lower level features in their model as well [Mostafazadeh et al., 2017]. The most compelling cases of this were Flor and Somasundaran’s *mflor* model which uses lexical cohesion and sentiment analysis trends [Flor and Somasundaran, 2017] and the top performer on the task, Schwartz’s *msap* model which uses predominantly language modeling, only applying linguistic features to the final sentence to capture stylistic biases of the

authors. In Flor and Somasundaran’s work, they were able to obtain 62% accuracy without recognizing any narrative, linguistic, or knowledge-based elements of the task, going so far as to mention that they made no attempt at narrative understanding or semantic reasoning whatsoever. Given the performance of these non-narrative models and the contributions of non-narrative features in the feature sets, the task curators are in the process of taking another look at the testing sets to re-design them and potentially the paradigm yet again with the hopes of making narrative understanding and knowledge-rich approaches a necessity [Sharma et al., 2018].

### 6.1.3 Similar Work

As we will be exploring the use of narrative analogy as a feature for the Story Cloze Task, it behooves us to examine what other techniques have been used for similar purposes in the past as well as how narrative analogy has been used as a feature for learning before.

Our approach of narrative comparison by entity-alignment based upon predicate semantic similarity shares a surface resemblance to some notable previous approaches. Most obviously, it follows in the footsteps of work done on probabilistic event schemas. One such example is Cheung et al. [Cheung et al., 2013] where events are constructed from clauses which are used in a sequential topic modeling fashion to assign the events to frames and transition between said frames in an attempt to maximize the probability of document generation from the distributions denoted by the probabilistic frames. Chambers presents another especially relevant example of a similar type using narrative chains to generate event schemas [Chambers, 2013]. In addition, our methodology also shares commonalities with many script learning approaches such as the combination of event chains in script learning [Schank and Abelson, 2008, Mooney and DeJong, 1985] to provide event based coherence [Pichotta and Mooney, 2014, Pichotta and Mooney, 2016, Chambers and Jurafsky, 2008, Chambers and Jurafsky, 2009, Jans et al., 2012, Orr et al., 2014]. What differs, beyond the means by which we describe event sequences, is that in our work the comparison performed is based off discovered analogies between these narratives.

Narrative analogy has been used in other narrative-based NLP tasks, but has not seen much use outside of fable/folktale classification [Finlayson, 2012, Elson, 2012]. While analogical methods have also been used extensively in story generation [Zhu and Ontanón, 2010], their use for NLP labeling and analytical tasks remains largely untapped. To best understand how analogy has been used, we can look at the fable/folktale focused tasks studied by Elson and Finlayson. Elson explored the use of SME (see Section 2.1) as a feature for fable similarity classification where the size of the discovered SME graph isomorphism corresponded to the quality of the mapping, finding that it produced the highest F-score-style composite of accuracy and

completeness of discovered similarities amongst all the features evaluated as rated by Mechanical Turk users. Finlayson made heavy use of SME to label Russian folktales with Propp’s morphology [Propp, 2010], finding success on the more frequent morphologies with F-scores around 0.8.

The successful application of narrative analogy to what Chatman would call story-based tasks [Chatman, 1980] provides hope that using narrative analogy on the story-based Story Cloze Task might also prove fruitful.

#### **6.1.4 The Dataset**

The dataset for the Story Cloze Task consists of three sets of stories: a training set, an evaluation set, and a testing set. The training set contains 98,099 five-sentence stories without any alternative narrative endings (does not have two possible 5<sup>th</sup> sentences). The evaluation set has 1,851 stories which have four fixed context sentences with two alternative story-ending 5<sup>th</sup> sentences with a label for which is the correct ending. Similarly, the testing set also has 1,851 stories with two alternative 5<sup>th</sup> sentences with the correct completion marked.

### **6.2 Analogical Methodology**

In order to present our own solution to the Story Cloze Task we need to devise a method of making the binary choice between narrative completions given some analogically informed measure of likelihood. We present a simple solution to this by means of analyzing how two copies of each evaluation/test case, one for each potential 5<sup>th</sup> sentence, relate to the training set of narratives.

To see how our two test cases interact with other narratives, we will look at the analogical similarity between the narratives which comprise them. Since our test cases may have multiple narratives, we need only consider those narratives on which they differ – only those narratives which contain at least one predicate in one of the 5<sup>th</sup> sentences. If we have more than one such narrative, we can simply average the analogical similarity over all present narratives since each one plays a role in the final sentences and thereby plays a role in distinguishing between the two candidate narrative completions. Before we can do any of this, we need to define a measure of analogical similarity.

#### **6.2.1 Similarity Function**

As was mentioned above, LISA has a built in measure of analogical similarity, however due to the computational expense of the LISA algorithm it is unsuitable for doing pair-wise comparison on large datasets.

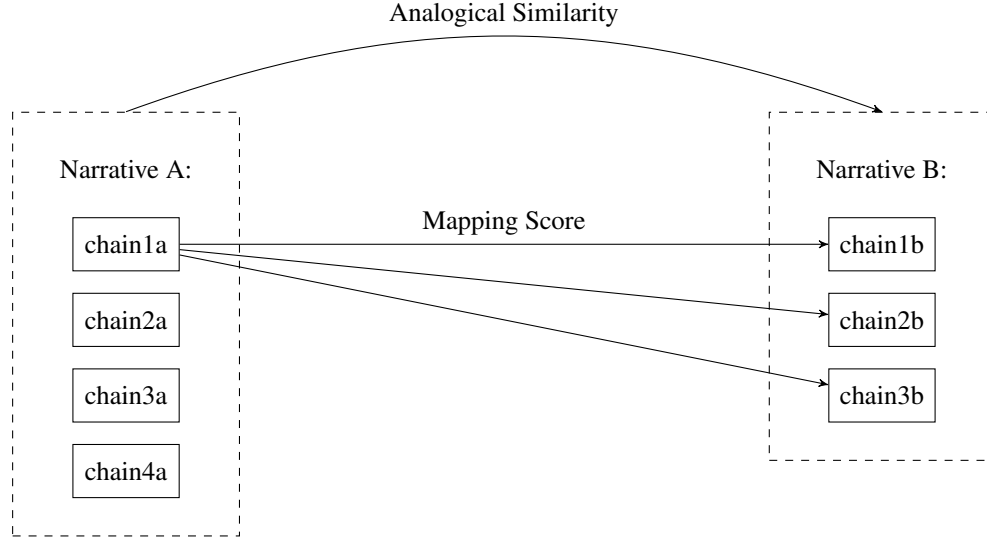


Figure 6.1: To understand mapping between narratives we look at mapping between the chains which comprise them. Every chain is compared to every other chain. Only the outgoing edges from *chain1a* are represented here to reduce clutter.

Instead, we developed a novel measure inspired by aspects of LISA’s recall and mapping. Our measure captures aspects of analogical similarity in that it attempts to find the mappings between objects which best align verb-roles belonging to their respective narrative chains such that the cosine similarity of the semantic representation of the verb-roles is maximized. That is to say, we find the object to object mappings which have the best alignment of verb-roles, assigning a measure of analogical similarity based upon the semantic similarity between said respective verb-roles. For example, given two narratives, “Narrative A” and “Narrative B”, we need to find a way to capture a measure of analogical similarity so that we can compute a transition matrix to use in MCL. Ultimately we need to fill in all such similarity scores between every narrative in our dataset. Since each narrative is composed of narrative chains, we can push the task of computing analogical similarity downwards to the level of narrative chains, over which we need only compute a score for mapping a single object onto another object. This breakdown can be seen in Figure 6.1.

We have reduced the problem to narrative chains, but we still need a way to calculate the mapping score over them. To get some ideas, we can look at how LISA handles mapping and generating its mapping scores. LISA’s mapping score is obtained by computing a full object and predicate matching such that it follows the restrictions laid out in Section 3.3. It calculates this by looking at the mapping weights assigned to its discovered mapping. This mapping discovery in turn is driven by finding analogical matches for LISA’s SP Units while enforcing consistency requirements. SP Units, a pairing unit between objects and verb-roles, are not as clear-cut in terms of narratives and narrative chains. Instead, since our measure of mapping score is driven by narrative chains, and thus only directly considers mapping objects, it is worthwhile to explore what

trade-offs this difference presents by looking at two benefits and two limitations of our approach:

1. Computational Complexity: This allows us to have a much simpler computational mapping space, and can be performed substantially faster.
2. Optimality: Since the full mapping has the potential for immense computational expense, LISA does not necessarily ensure that the discovered mapping is optimal, but instead ensure local optimality at each step of the process. Restricting ourselves to calculating mappings on objects and verb-roles allows us to ensure optimality in a reasonable computational time at the expense of guaranteed consistency.
3. Consistency: By restricting the mapping to object-to-object and verb-role-to-verb-role mappings we lose out on the guarantee of one-to-one verb mapping (two verb-roles from the same verb could map onto verb-roles belonging to two different verbs). in principle we should often preserve consistency by virtue of object and semantic alignment, but since it is not explicitly constrained, degenerative mappings may occur.
4. Recursive Structure: LISA uses Child Props to allow Proposition units to take other Proposition units as arguments. Since we have restricted our consideration to an object-centric mapping, these propositional arguments do not appear at all. That is to say, the verb-role which is bound to a proposition does not show up in any narrative chain since there is no object associated with it, thus limiting our representational capacity.

As a whole, this method brings in elements of LISA's recall and mapping algorithms. LISA drives recall by simply comparing the joint semantics of bound object and verb-role pairs, while full-blown mapping on the other hand requires parallel constraint optimization to try to find structurally consistent and semantically optimal mappings. Our proposed mapping paradigm bases itself loosely on the recall aspects of LISA, while at the same time requiring structural consistency within object units. If we wanted to rank it in terms of 'analogical correctness' our object-centric mapping should sit somewhere above LISA's recall but below its full-blown mapping.

In order to calculate the object-to-object mapping scores we can once more break down each narrative chain into its constituent verb-roles as in Figure 6.2. Now the issue of calculating similarity has become a semantic comparison task for which we have a wealth of techniques to solve; the simplest and most sensible of which being cosine similarity. To establish a mapping score, we need to compute all pair-wise cosine similarity measures on verb roles between two narrative chains. For explanation purposes, we can look at an imagined scenario as given by Table 6.2. In this matrix, every element corresponds to a potential mapping between two verb-roles as given by its indices. As such, the task of finding a mapping can be

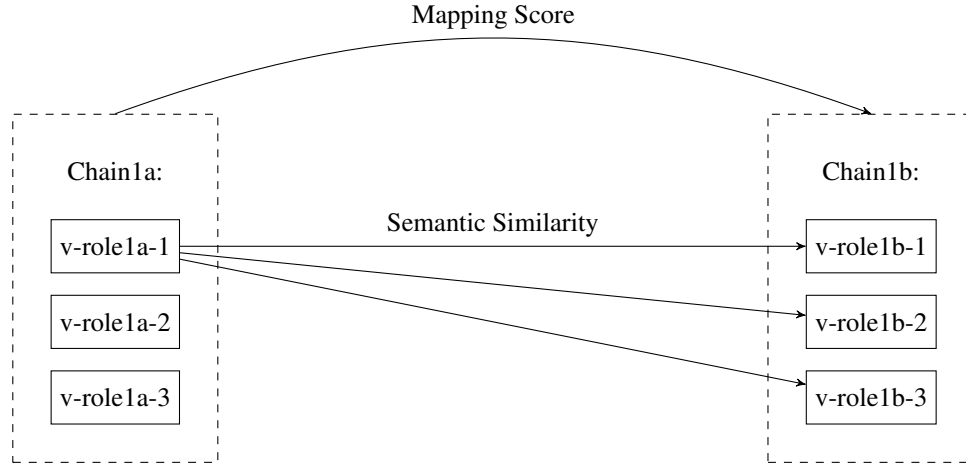


Figure 6.2: Similar to Figure 6.1, to figure out a mapping score between narrative chains, we breakdown the chains into verb-roles and compare mappings between them, thus reducing the problem to a semantic comparison task.

	v-role1b-1	v-role1b-2	v-role1b-3
v-role1a-1	0.7	0.2	-0.1
v-role1a-2	-0.3	-0.2	0.6
v-role1a-3	0.1	0.8	0.7

Table 6.2: An example of a potential pairwise similarity matrix on verb-roles. Shaded entries correspond to the selections for an ideal mapping.

viewed as selecting a subset of elements of the matrix which correspond to the best possible mapping. Since we know that each verb-role will only map to one verb-role, in order to maximize the semantic similarity and get a semantically ideal map, we want to select the largest elements of our similarity matrix such that no two elements share a row or a column. That is, any two elements which share a row or column would, when selected jointly, represent a verb-role mapping to two different verb-roles. This violates the one-to-one constraint of analogical mapping and thus should not be considered. Our task of finding an ideal mapping can then be reduced to finding the set of matrix elements such that their sum is maximized and no two elements share a column or row. In Table 6.2 the shaded values have the maximal sum without overlapping columns or rows, and thus correspond the ideal mapping between their corresponding narrative chains.

In the case of Table 6.2 with its largely straightforward similarity values and small size, computing which subset yields the maximal sum without violating one-to-one constraints is straightforward. But, as chains become larger this task can become substantially more complex. To avoid running into the same computational expense problem which LISA has, we need a means to cheaply and quickly find the maximal subset. To see exactly how necessary this is, if we were to brute force the solution, looking at every subset of an  $n$  by  $n$  matrix (while in principle these matrices are often not square, for ease of computation we will explore

the square case), we would need  $\mathcal{O}(2^{n^2})$  operations. By restricting first over the row-column uniqueness constraint, we could reduce that to  $\mathcal{O}(n!)$  which by Stirling’s Approximation is roughly  $\mathcal{O}((n/e)^n)$  which is still untenable for computation on large datasets with narrative chains of significant size. In order to bring this down to polynomial time – preferably a small order polynomial time – we need a different algorithm.

The Hungarian Method [Kuhn, 1955] is a polynomial time algorithm for solving the combinatorial problem called the “Assignment Problem” wherein we are presented with a complete weighted bipartite graph (a graph whose vertices belong to one of two groupings where each vertex is connected to all vertices in whichever group to which it does not belong and none from its own group) and asked to select edges such that each vertex has exactly one edge attached to it from the selected edges and the total sum of all weights in the selection is minimized. The Assignment Problem is very similar to our task in that each verb-role can be considered a vertex in the graph, and each edge a potential mapping. Using this correspondence, we have an inherent bipartite structure since each verb-role has a similarity score to every verb-role in the narrative chain to which it does not belong. The only way in which the two setups differ is that the Hungarian Method minimizes edge weight sum whereas we would like to maximize it. A simple change to the cost function used in the Hungarian Method, or if we would prefer, altering the edge weights to be negative results in the minimization the algorithm typically performs becoming a maximization task and thereby viable for our calculations. The latest form of the Hungarian method runs in  $\mathcal{O}(n^3)$  [Jonker and Volgenant, 1987] making it far more tractable, and suitable for bulk use over larger narrative chains in large dataset situations. Using our modified Hungarian Algorithm we can compute which elements of any given similarity matrix correspond to an ideal mapping, allowing us to then use the similarity scores of the ideal map to compute a mapping score between the narrative chains as a whole. Given that the Hungarian Method yields a set of elements from the semantic similarity matrix, call it  $S_c$ , we can compute a mapping score between narrative chains using Equation 6.1.

$$\phi(c1, c2) = \frac{\sum_{e \in S_c} e}{|c1|} \quad (6.1)$$

In Equation 6.1 we have a mapping function which takes in two narrative chains,  $c1$  and  $c2$  and returns the normalized sum of the entries the Hungarian Method returns from finding the maximal mapping over  $c1$  and  $c2$ ’s similarity matrix. The normalization is done by the total length of  $c1$  – how many verb-roles  $c1$  contains. The use of  $c1$  here is what makes the mapping score asymmetric since the actual semantic similarity, and thus the result of the Hungarian Method, are all symmetric. This asymmetry is quite natural in that it is directly born from the notion that given two similar stories, the smaller one should map well onto the larger one, but not vice-versa. Or, more theoretically, the inference and reasoning capabilities of an analogical system are

	Chain1b	Chain2b	Chain3b
Chain1a	0.7	-0.8	-0.1
Chain2a	0.1	0.8	0.7
Chain3a	-0.6	0.1	0.5
Chain4a	0.2	-0.3	0.3

Table 6.3: An example of a potential pairwise mapping-scores matrix over narrative chains. Shaded entries correspond to the selections for an ideal object alignment.

better facilitated by seeing how a smaller piece of a narrative fits into a larger one rather than the other way around.

With  $\phi$  defined, we can now tackle generating a measure of analogical similarity as shown in Figure 6.1 using the same Hungarian Method where the similarity values in our matrix are the mapping scores between narrative chains as given by  $\phi$ . For the mapping score between ‘Chain1a’ and ‘Chain1b’ of the example given by Table 6.2, we can take the results of the Hungarian Method, namely  $S_c = (0.7, 0.8, 0.6)$  and run  $\phi$  over it:

$$\phi(\text{Chain1a}, \text{Chain1b}) = \frac{0.7 + 0.8 + 0.6}{3} = 0.7$$

Continuing the example, we can produce a matrix of mapping scores between narrative chains where each mapping score is calculated using  $\phi$  such as the one in Table 6.3. To figure out the optimal object-to-object mapping between ‘Narrative A’ and ‘Narrative B’, we need to figure out the same optimization issue as before over verb-roles. To solve this, we can again turn to the Hungarian Method where we now have narrative chains as the vertices and mapping scores as the edge weights of our complete bipartite graph. Solving for the optimal alignment, we can again get a subset of matrix elements whose values have a maximal sum without violating the one-to-one constraints of mapping – the shaded entries of Table 6.3. So, with an identified set of entries, call it  $S_n$ , we can compute a measure of analogical similarity between narratives in much the same way as our mapping score  $\phi$ :

$$\Phi(N_A, N_B) = \frac{\sum_{e \in S_n} e}{|N_A|} \quad (6.2)$$

Here we take two narratives  $N_A$  and  $N_B$  and return the sum of the Hungarian Method returned matrix cells, which correspond to the ideal mappings between narrative chains, normalized by the number of narrative chains in  $N_A$ . With  $\Phi$  we now have a way to calculate narrative similarity.

To better understand the process, it may be beneficial to see it happen with an example. As before, we will turn to the two stories concerning Sam and Bob: “Sam cooked a steak. The steak rested. Sam ate the steak.” and “Bob bakes a cake. The cake cools.” and calculate the analogical similarity from Sam’s narrative



to Bob's. The first step is to represent the two stories as narratives:

*NarA* :

*chain1a-Sam* : *cooker, eater*

*chain2a-steak* : *cooked, rested, eaten*

*NarB* :

*chain1b-Bob* : *baker*

*chain2b-cake* : *baked, cooled*

So, to calculate the analogical similarity from *NarA* to *NarB* we first have to calculate the mapping scores from each of *NarA*'s two chains to each of *NarB*'s two chains, yielding a two-by-two matrix of similarity scores. In turn, to calculate the mapping score for each cell of the two-by-two matrix we have to calculate the semantic similarity between each verb-role of the two chains in question. So, first we figure out the mapping score from *chain1a-Sam* to *chain1b-Bob* which requires we fill in the two-by-one matrix of semantic similarity as follows:

	baker
cooker	0.9
eater	0.4

This allows us to see that the optimal mapping is from *cooker*  $\leftrightarrow$  *baker*. However, since there are two verb-roles in *chain1a-Sam* and only one of them maps successfully, the actual mapping score from *chain1a-Sam* to *chain1b-Bob* is only one half the cell value of 0.9, giving us an effective mapping score of  $\frac{0.9}{2} = 0.45$ . Next, we would need to find the mapping score from *chain1a-Sam* to *chain2b-cake* which would yield the following matrix:

	baked	cooled
cooker	0.1	-0.1
eater	0.2	-0.2

Which gives us the optimal mapping *cooker*  $\leftrightarrow$  *cooled* and *eater*  $\leftrightarrow$  *baked* a mapping score of  $\frac{-0.1+0.2}{2} =$

0.05. Here we list the optimal mapping, even though it is a rather poor mapping, because we will need this calculation to ultimately calculate the analogical similarity between  $NarA$  and  $NarB$ .

Now we move on to considering *chain2a-steak*'s mappings to  $NarB$ 's chains. First we consider the mapping from *chain2a-steak* to *chain1b-Bob*:

	baker
cooked	0.1
rested	0.4
eaten	-0.2

Which gives us the optimal mapping of *rested*  $\leftrightarrow$  *baker* with a mapping score of  $\frac{0.4}{3} = 0.13$ . Lastly, we look at the mapping from *chain2a-steak* to *chain2b-cake*:

	baked	cooled
cooked	0.9	0.4
rested	0.2	0.6
eaten	0.6	0.4

Which yields the mappings *cooked*  $\leftrightarrow$  *baked* and *rested*  $\leftrightarrow$  *cooled* with a mapping score of  $\frac{0.9+0.6}{3} = 0.5$ .

Using all these calculated mappings, we can fill in the chain-to-chain comparison matrix to calculate the total analogical similarity:

	chain1b-Bob	chain2b-cake
chain1a-Sam	0.45	0.05
chain2a-steak	0.13	0.5

Using this, we find the ideal mappings of *Sam*  $\rightarrow$  *Bob* and *steak*  $\rightarrow$  *cake*<sup>1</sup> with an analogical similarity of  $\frac{0.45+0.5}{2} = 0.475$ .

Now that we understand how the scores are calculated, we can use the measure of analogical similarity as a feature for learning or a measure for clustering.

<sup>1</sup>While the ideal mappings are symmetric here, this is not guaranteed since chain length asymmetry introduces a different denominator in the mapping score calculations.

## 6.3 Narrative Analogical Clustering

The hope of clustering of narratives is that by producing analogical groupings we get a proxy for schematization. Here we want to explore if these induced clusters provide any meaningful improvement to the Story Cloze Task in the hopes that we can better understand their utility towards story-based tasks at large.

While narrative event understanding has long been recognized as an important part of natural language understanding (NLU), previous approaches to it have mostly centered on the development of probabilistic scripts or schemas [Pichotta and Mooney, 2016, Pichotta and Mooney, 2014, Chambers, 2011, Chambers, 2013] rather than focusing on possible analogies between narratives to generate them as we will examine below. Analogy based NLP or NLP-adjacent work has instead either been reliant on hand-encoded representations [Elson, 2012] or been focused on more niche tasks instead of narrative comprehension or other aspects of NLU [Baydin et al., 2015, Reiter, 2014]. More salient is some analogy related work, specifically with SME, which has been done using event-by-event alignment to try to identify narrative morphological units and overarching themes such as a struggle/victory, rescue, or reward [Finlayson, 2012]. However, none of these works attempt to put forth a method which allows for explicit mapping of full analogies towards any form of inference or reasoning.

Our specific task will be to identify a means of narrative analogical clustering analysis which can potentially lend itself to a variety of NLU applications. In order to make this analysis more approachable for general machine learning, we propose a method of measuring analogical similarity that is informed by aspects of the LISA model of analogy and then use this measure to perform standard clustering over the Story Cloze training set of open domain natural language stories.

### 6.3.1 Previous Uses of Event Clustering

Narrative clustering is a direct extension of previous work mostly centered around event clustering. Event clustering has shown great promise across a variety of tasks from schematization/script learning to relation detection [Chambers and Jurafsky, 2008, Brody, 2007, Orr et al., 2014]. The step of moving this kind of analysis to collections of events, such as those forming a narrative, is a natural one. This notion is further supported by the work of Chambers and Jurafsky [Chambers and Jurafsky, 2009] focused on clustering events to generate script-like narratives ultimately comparing favorably to FrameNet’s frame structures [Baker et al., 1998]. These discovered script-like narrative structures are ultimately similar in spirit to what we will explore clustering – Chambers and Jurafsky’s script-like structures are built by selecting events whose roles are deemed sufficiently similar to the roles of other events already in the ‘script’, whereas ours are generated

from events found in single examples. Similarly verb clustering using WordNet [Miller, 1995] relations has been used to generate FrameNet’s event clusters as well [Green and Dorr, 2005]. Based upon this body of literature, it seems worthwhile explore clustering of complete narratives.

In order to analogically cluster full narratives instead of the events which comprise them, we need several functioning pieces working in concert with one another. First, we need a means of representing narratives that lends itself to clustering. Second, we need a way to measure the analogical distance or similarity between the narrative representations. Third, we need to choose a clustering algorithm which is sensible for our dataset. And lastly, we need a means to evaluate our clusters. Below we will examine each of these steps and the approach we used and why we used it.

### 6.3.2 Narrative Structure

For narrative representation, we can draw off of our previous work on automating the LISA model of analogy. However, drawing directly from LISAese produces some notable issues. LISA’s structure is intentionally recursive and referential which makes simple calculations for clustering difficult. Instead, we’d like to devise a representation which facilitates cheap and easy comparison at the expense of neural plausibility and perhaps some analogical power.

The approach we will follow is one which, much like our automated LISAese production algorithm, is based upon Chambers and Jurafsky’s Narrative Chains [Chambers and Jurafsky, 2008, Chambers and Jurafsky, 2009]. These chains follow actors through the story, tracking the verb-roles they fill (verb dependencies in Chambers and Jurafsky’s work) which we determine through Semantic Role Labeling and subsequently attach overlapping chains into full narratives<sup>2</sup>.

Each narrative is then defined as a collection of narrative chains which transitively overlap each other on at least one verb. That is, for every pair of chains in the narrative, they either contain two verb-roles of the same verb or are similarly linked by a third (or more) chains such that the two chains have a verb-role overlap with their linking chain. For example, we can look to the story “Bob baked a cake. The cake cooled. Sally ate the cake.” which has three chains following the nouns “Bob”, “cake”, and “Sally”. The story forms a single cohesive narrative since the all three chains are connected via overlapping verb-roles. The “Bob” and “cake” chains overlap on the verb “baked”, the “cake” and “Sally” chains overlap on the verb “ate”, and while the “Bob” and “Sally” chains don’t directly overlap, by following their mutually overlapping third chain – namely the “cake” chain – they have transitive overlap, thus forming a single cohesive narrative.

In summary, a narrative is a set of narrative chains which in turn are sets of verb-roles. This structure al-

---

<sup>2</sup>This process is covered in more depth in Chapter 5

lows a multi-level hierarchy of comparisons where we can base the similarity of each level on the similarities of the level below it, resulting in only needing a means of direct comparison of verb-roles.

### 6.3.3 Verb Semantics

In order to compare verb-roles, we need to semantically ground them. To establish this grounding, we explored three forms of verb-role semantic embedding:

1. Plain Words: Every verb-role is represented by the GoogleNews pre-trained Word2Vec [Mikolov et al., 2013b, Mikolov et al., 2013a, Mikolov et al., 2013c] embedding of its base verb. For instance both the verb-roles for lover and beloved (*love\_A0* and *love\_A1*) would be semantically encoded with the Word2Vec embedding for “love”. The simplest semantic embedding can be formed by ignoring the difference between roles and using only the semantics of the verb itself. We can treat this as a simple base-line to which we can compare the other two semantic methods. There are numerous theoretical issues with this technique; and it misses a wealth of narrative information. Nonetheless, it may be useful to see the impact that role-based semantics has on the process of analogical clustering.
2. Co-reference Chains: Each verb-role is represented independently of its base verb. The embeddings are formed by running Word2Vec over narrative chains (which consist of verb-roles filled by the same actor in a given story).
3. PropBank Frames: Verb-roles are represented by the GoogleNews pre-trained Word2Vec semantic embedding of the role description listed in PropBank’s Frame files for the given role.

The constructions of both ‘Co-reference Chain’ and ‘PropBank Frame’ based embeddings are covered in greater detail in Subsection 5.2.2.

In order to facilitate comparison with LISA, which at the time used sparse semantic encodings, we sparsified all the representations using Faruqui et al.’s word vector sparsification technique [Faruqui et al., 2015] to increase the 300 dimensional word embeddings to a 3000 dimension space in which the vectors were over 90% sparse

With structural and semantic representations determined, we are left with the task of clustering itself. The goal of clustering here is to associate narratives which are analogically similar to one another. To accomplish this we must use our measure of analogical similarity as well as choose a clustering algorithm with which to associate the narratives.

### 6.3.4 Clustering Techniques

The clustering techniques, for the purposes of our consideration, fall most naturally into one of two main categories: Distance Based or N-Dimensional Embedding Based. What we mean by this is that the clustering algorithms either need a distance function between any two narratives, or a meaningful way to embed a given narrative into some fixed n-dimensional space. Many popular clustering techniques such as K-means [Lloyd, 1982], spectral clustering [Ng et al., 2002], mean-shift [Cheng, 1995], and Gaussian mixture models all have the expectation that data-points lie in some fixed dimension vector-space [Xu and Wunsch, 2005]. Since the size of each narrative chain and the total number of narrative chains are highly variable across narratives, dimensional embedding becomes quite difficult, requiring an arbitrary length input to be embedded into a common n-space, perhaps by a recursive neural network (such as an LSTM) or any other form of sequence embedding. This is further compounded by the fact that we want our n-space to allow direct comparisons of unordered verb-roles and unordered narrative chains since the chains and narratives do not encode any notion of temporality. It is however much easier to establish a straightforward distance metric between two narratives. Indeed, we already have a measure of analogical similarity in the form of LISA’s mapping score. Unfortunately the calculation of LISA’s score via analogy discovery is incredibly computationally expensive and prohibitive for comparison between all narratives on any large dataset. Instead, we need to define a new, faster, distance function which we will discuss below.

Before we decide on a distance function, we must first decide upon a distance based clustering technique. Many connectivity based clustering techniques require that the distance used obey metric axioms. However, analogical similarity often violates these constraints, most notably in symmetry and triangle-inequality. Symmetry here means that given two set members,  $A$  and  $B$ ,  $A$  is exactly as similar to  $B$  as  $B$  is to  $A$ . As we will argue the benefits for later, we can give up symmetry violation for the purposes of clustering. But the violation to triangle-inequality proves to be much more fundamental to analogy. The triangle-inequality is a metric property which states that for any three elements of our set,  $A$ ,  $B$ , and  $C$ , the distance between  $A$  and  $C$  must be less than the sum of the distances between  $A$  and  $B$  plus  $B$  and  $C$ . To see how analogy violates this, we can look at a classic example from Tversky of psychologically valid analogies: Jamaica is analogically similar to Cuba, and Cuba is analogically similar to Russia, but Jamaica and Russia are dissimilar [Tversky, 1977] and thus analogically far apart. While this is on word analogies, the same issue holds true on analogies over narratives since the aspects of the stories which inform initial mappings and ground the two ‘good’ analogies can be along totally disparate elements while still preserving structurally sound, coherent, and useful analogies. For example, if we have three stories, one about making a purchase, another about going to a new restaurant, and the third about taking a trip we have the same situation as in Tversky’s

example. The purchase can have significant overlap with the aspect of buying food at a restaurant, and going to a restaurant can include both the travel to the restaurant and new experiences of different cuisine much like travel to a new location. However, making a purchase and traveling to a new location could likely have little to nothing in common, violating the triangle inequality. As a result, if we want our clustering algorithm to be able to take advantage of analogical similarity, it can't depend upon a metric distance or similarity function.

Beyond metric axiom violations, the way that we define narratives results in the existence of what we call "proto-narratives". Proto-narratives are small narratives, often only one or two propositions in length, which map well into many disparate stories, but are in general not mapped well onto. For example, in the following story:

For our first date, Sam and I went for a walk. It turned out to be a beautiful day. In the end, we stayed outside all morning. We had so much fun that we will probably have a second date.

In this story we have two narratives:

*Narrative1 :*

*go(Sam\_and\_I, for\_a\_walk, for\_our\_first\_date)*  
*stay(Sam\_and\_I, outside, all\_morning)*  
*cause(have(Sam\_and\_I, so\_much\_fun), have(Sam\_and\_I, probably, second\_date))*

*Narrative2 :*

*turn(it, to\_be\_a\_beautiful\_day)*

Here *Narrative2* is a proto-narrative and as we can see it doesn't really contain a meaningful information for schema generation, analogical reasoning, or inference nor does it have any potential for improving a meaningful narrative cluster. It does however have a high likelihood of being 'analogically close' to many meaningful narratives since the semantics of *turn(it, to\_be\_a\_beautiful\_day)* can find many potential "good" mappings onto analogs so long as it matches some proposition in the story since the structural match restrictions of forming a good analogy are non-existent with only a single proposition to consider. This makes proto-narratives a serious concern for many clustering techniques since they could likely be highly connected to many otherwise unrelated narratives. To combat this, we need a clustering algorithm that doesn't consider only raw distance/similarity scores. For example, since proto-narratives are good matches to so many narratives, if we instead take a weighted average across all outgoing similarity scores to generate a normalized

distance/similarity score we can remove the impact of these proto-narratives, but our clustering algorithm must be able to adequately deal with probabilities instead of absolute distances. To summarize, we need a clustering algorithm which uses normalized distance/similarity scores, that does not expect those scores to follow metric axioms, and which does not expect an embedding into a fixed  $n$ -dimensional space. Exploring various clustering algorithms [Chen and Ji, 2010, Schaeffer, 2007], we find that Markovian Clustering (MCL) meets all our requirements [Van Dongen, 2000]. While MCL has seen use in NLP [Dorow et al., 2004], it is extensively used in the fields of genomics and bioinformatics [Qin et al., 2010, Li et al., 2003, Enright et al., 2002, Dunn et al., 2008, Karlsson et al., 2013].

Markovian Clustering is a graph-based clustering algorithm that follows a simple progression of matrix transformations on a transition matrix, which is a graph’s adjacency matrix – traditionally column-major for MCL – where each column (or row in row-major instances) is a probability distribution for the likelihood of a random walk following any given outgoing edge. The basic steps of MCL are as follows:

1. Expansion Step: The expansion step replaces the transition matrix with one describing the probability of where a random walk will end up after  $n$  steps ( $n$  here is a model parameter). To get this, we simply take the original transition matrix to the  $n^{th}$  power. In actual usage,  $n$  is often set to 2 and indeed many implementations fix this expansion parameter at 2.
2. Pruning Step: After squaring (or taking to a higher power) the transition matrix, many small probability transitions are often introduced. To combat this hit to performance there is optionally a pruning step here where columns are often pruned to either remove all but the top  $n$  values or to remove any value below some threshold.
3. Inflation Step: To push the transition matrix towards higher probability transitions, each entry in the matrix is replaced with itself taken to some power  $i$ . As a result, high probability transitions are minimally affected while low probability transitions are heavily penalized. Typically, meaningful  $i$  values lie somewhere in the range  $[1.2, 5]$  with higher inflation parameters corresponding to more fine-grained (and more numerous) clusters.
4. Normalization Step: Before we can repeat the original Expansion Step, we need to re-normalize our columns since the Inflation Step results in a non-stochastic matrix, and thus is not a viable transition matrix. And so, each column is normalized divisively.
5. Convergence Check: MCL will continue following the above steps until the changes they produce in the transition matrix settle. There are several ways to determine convergence, a couple frequent and simple conditions are that the maximum change to any entry is below a fixed threshold or perhaps that



the sum of the squares of all changes to entries is below a fixed threshold. If the change to the transition matrix meets the convergence criterion, MCL halts, otherwise the process repeats.

With a clustering algorithm decided on, we can now use our measure of analogical similarity to produce a transition probability matrix for the similarity graph of our narratives.

### 6.3.5 Markovian Clustering

To get the necessary input for MCL, we need to label our edges (and thereby our transition matrix) with a probability of transition. This can quite simply be done by normalizing over outgoing edge similarity measures, and since MCL is traditionally column-major, we normalize over columns of our narrative similarity matrix to get a viable transition matrix.

As far as edge symmetry goes, MCL only requires that for every entry in the transition matrix  $a_{i,j} \neq 0$ , we also have  $a_{j,i} \neq 0$ . This requirement is guaranteed by the nature of the computation of  $\Phi$  since the only element which changes between  $\Phi(N_A, N_B)$  and  $\Phi(N_B, N_A)$  is the denominator. However, such differences are not necessarily useful since a large difference between  $\Phi(N_A, N_B)$  and  $\Phi(N_B, N_A)$  would have a deleterious effect on mapping performance, and would serve no useful analogical function. As mentioned before, while asymmetry of mapping score is useful for analogical analysis, reasoning, and inference, this difference is less useful for clustering. If there is a large dissimilarity between transition probabilities, the generated clusters will still likely place them together. This happens since cluster membership is a binary choice and the high-outgoing-transition-probability narrative will end up having a high transition probability to whatever cluster the other narrative belongs to. Indeed, since most of the high outgoing-transition-probability narrative's probability weight flows through the other narrative anyway, the asymmetry serves no analogically useful purpose. It does however cost a great deal of computational time since, in general, the greater the asymmetry the longer the clustering process will take to settle. In order to avoid this, we must symmetricize the matrix which we did by taking the average of the transition matrix  $A$  and its transpose  $A^T$ . It is important to note that the symmetricization step must be done on the transition matrix (after column normalization) rather than on the matrix of analogical similarities or else our resulting transition probabilities could easily become dominated by proto-narratives which have high analogical similarity to many disparate narratives. By first normalizing, narratives with many high outgoing edge weights end up with only low transition probabilities which won't overwhelm their neighbors. After symmetricizing the transition matrix, we must again column normalize it to ensure the matrix remains stochastic and can then begin the task of clustering.

## HipMCL

In order to ensure a reasonable run time and per-node memory requirements, we needed a highly parallel sparse MCL implementation. HipMCL [Azad et al., 2018] is just such an implementation, as it is highly optimized for sparse workloads and massively parallel clusters. HipMCL has largely been used in the BioIn-formatics domain; however, as the clustering task itself is universal, we will use it here.

The MCL algorithm leaves open the question of how to determine convergence of the clustering, with many implementations requiring that all changes to the transition matrix be below some fixed threshold for one iteration. HipMCL takes a slightly different approach. It defines convergence by a calculation of the “chaos” of the system which is calculated as the maximum across every column of the number of non-zero elements times the maximum value in that column minus the sum of the squares of the whole column. Equation 6.3 gives the mathematical definition of HipMCL’s “chaos” function.

$$chaos(A) = \max_j ((\max_i (a_{ij}) - \sum_i (a_{ij}^2)) \cdot nnz_j) \quad (6.3)$$

However, as defined, “chaos” here is insufficient for our needs. Many narratives could ride the cusp between two clusters, and so enforcing a hard-clustering requirement that the number of non-zero elements in a column must be 1 (i.e.  $nnz_j = 1$ ) might ensure that the “chaos” of our transition matrix never approaches zero and thus HipMCL never finishes. To remedy this, we simply alter the default HipMCL “chaos” function to not contain the  $nnz_j$  term, giving us a new convergence function as shown in Equation 6.4

$$chaos'(A) = \max_j (\max_i (a_{ij}) - \sum_i (a_{ij}^2)) \quad (6.4)$$

This change in the “chaos” function allows stories to belong to multiple clusters. However, the MCL algorithm is not natively a soft-clustering algorithm, and will try to push stories into exactly one cluster. Indeed, the new measure of “chaos” will only reach zero if the all the remaining non-zero elements of a column in the transition matrix are exactly the same which would only happen when MCL determines that a narrative belongs exactly equally well to multiple clusters. If there is any preference for one cluster over another, MCL will cluster narrative exclusively with the preferred cluster. If we weren’t to allow the model to place narratives into multiple clusters like this, the only other recourse would be to collapse all the clusters to which the narrative has equal membership into a single larger cluster. This practice falls prey to proto-narratives which can frequently map with exactly the same, namely perfect, analogical similarity onto several otherwise dissimilar stories. Since these proto-narratives can form bridges between two or more unrelated clusters, col-

lapsing along them can result in poor cluster cohesion.

Given that this is a departure from HipMCL’s convergence criteria, it is worthwhile to note that our new “chaos” function will always converge whenever the original would. This is easy to see since our definition is strictly looser, given that  $nnz_j \geq 1 \rightarrow chaos(A) \geq chaos'(A)$ . And thus we can operate comfortably knowing that our new convergence criteria will not result in infinite clustering time.

Using the new “chaos” function we have a working means of clustering narratives based off of a LISA-inspired computationally feasible measure of analogical similarity.

## 6.4 Constructing Features

Now that we have an analogical similarity measure and clustering method defined, we can begin producing features for the Story Cloze Task. The first set of features we will explore are based directly off of the measure of analogical similarity. These can be constructed in a straightforward fashion by considering best and average similarities across the whole of the training set. Defining how a story interacts with a cluster of narratives is less straightforward as there are many potential measures such as a cluster’s cohesiveness or perhaps the distance to some identified cluster centroid. Instead, we turned to a simpler approach, namely “closeness centrality”, a measure of how central a single node is to a given cluster [Bavelas, 1950]. Closeness centrality is calculated as the inverse of the average distance between a node and every element of the cluster as shown in Equation 6.5.

$$C(x, X) = \frac{|X|}{\sum_{y \in X} d(x, y)} \quad (6.5)$$

In Equation 6.5  $C$  is the closeness centrality of a narrative  $x$  and a cluster  $X$  and  $d(x, y)$  is the distance between  $x$  to  $y$  (shortest path distance in the case of a non-complete graph). However, since the measure we have defined is one of analogical similarity rather than distance, we need to use a different measure rather than closeness centrality. The simplest and most straightforward way is to simply invert it such that we define a new closeness measure, call it  $C'$  such that  $C'(x, X) = C(x, X)^{-1}$ . Using this we can easily calculate how close to our discovered clusters the competing narrative completions make the stories as a whole.

So, we now have the means to capture both how testing narratives interact with individual training narratives as well as how they interact with the clusters as a whole. These measures can be broken down into four cases:

1. Largest Cluster Centrality: The single best  $C'$  measure found across all clusters.
2. Average Cluster Centrality: The average of the  $C'$  measures across all clusters.

3. Largest Individual Similarity: The training example with the single largest narrative similarity.
4. Average Individual Similarity: The average narrative similarity over all training examples.

Since our measure of analogical similarity is asymmetric, we double these four cases by considering both incoming and outgoing measures to give us a features set of eight analogically motivated values. These eight values correspond to features for a single story completion (one of the two possible story endings). So, each story completion produces an 8 dimensional vector, with the final classification determined by which vector is a better fit for the “correct” label as assigned by a classifier. That is, we classify each completed version of the story and choose whichever one the classifier likes better. In the event that the classifier gives the same probability weight, we randomly select the story completion. To make our analysis less stochastic, when calculating accuracy we fix these even-matches with a score of +0.5 to the total number correctly selected since this best reflects the random baseline for the selection.

As we want to be able to cleanly analyze the impact of our features, we chose to use a simple Support Vector Machine (SVM) classifier from the scikit-learn package [Pedregosa et al., 2011] and train it over the evaluation set before getting performance results on the testing set. We will also explore the uses of various kernels, but suspect that a linear kernel will provide the greatest utility. We anticipate this since we would expect that high ‘Largest Individual’ scores will be especially meaningful, but proto-narratives will always have high, and largely meaningless, ‘Largest Individual’ scores. However, they will be the only cases which **also** have nearly identical ‘Average Individual’ scores. This pairing should allow the SVM to rapidly pick out these unwanted cases with a linear kernel, and while other kernels should be able to discover related decision planes, a linear kernel should do so with ease.

### 6.4.1 Clustering for Story Cloze

In order to obtain the features for our SVM, we have to implement our clustering technique and cluster the Story Cloze training set. For this, we turn the training set into a collection of narratives over which we can cluster. Doing this yields us 225,881 narratives from 98,099 total stories. Since Markovian Clustering uses matrix multiplication in its expansion step, we can expect a time complexity for this of  $\mathcal{O}(n^3)^3$ . Given this computational complexity, with a dimension of 225,881, we are looking at  $1.15 \cdot 10^{16}$  multiplications for each round of MCL as well as a memory requirement of 204 GB to simply store a single copy of the matrix in memory – assuming that transition probabilities are encoded as floats, if they are encoded as doubles, the memory requirements are 408 GB. As this was not feasible, we were forced to substantially sparsify our input.

---

<sup>3</sup>HipMCL does not use Strassen’s Algorithm

	PropBank	Co-Ref Chains	Plain Words
$i = 2.0$	3083	3498	528
$i = 2.25$	11281	10675	2586
$i = 2.5$	23262	22204	9151
$i = 2.75$	35746	34784	18746
$i = 3.0$	46896	45891	29319

Table 6.4: Number of clusters found for different inflation parameters using the three semantic representations given in Subsection 6.3.3.

The simplest way to sparsify our matrix would be to take only some top percentage of entries. If this sparsification were done over raw similarity scores, before normalization to transition probabilities, this would give an undue preference towards proto-narratives which have a high similarity to many narratives, so sparsification must happen after the normalization of the matrix. However, if we were to simply select the largest entries after normalization we would then be preferentially selecting for edges from sparsely connected nodes. That is, narratives which only map onto a few select narratives would have edges with larger transition probabilities, and thus dominate the sparsified matrix. To avoid this, we sparsified the matrix on a column-by-column basis<sup>4</sup>. Since the sparsification was performed within the context of a single narrative, this ensured that each node kept its top outgoing edges. In this way we reduced each column to its top 226 transition probabilities. To keep the size of the matrix within scope during clustering we set the pruning limits during MCL’s pruning step to keep the non-zeros in each column to at most 226 entries.

The other consideration is what inflation parameter MCL should use. The higher the inflation parameter, the more specific and thereby numerous the clusters are. To see the effect of different cluster sizes on the Story Cloze task, we ran a collection of five different inflation parameters from  $i = 2.0$  to  $i = 3.0$  at increments of 0.25. Running MCL over sparsified inputs for the three verb-role semantic embeddings described in Subsection 6.3.3 we get clusters as shown in Table 6.4. Using these clusters, we can now compute our features of and train an SVM on the Story Cloze evaluation set. We will repeat this process a total of fifteen times, one for each of the semantic embedding techniques paired with each inflation parameter.

## 6.5 Results

The results of the trained SVM on the Story Cloze Task can be seen in Tables 6.5, 6.6, and 6.7 for PropBank frames’ role description, narrative chain co-occurrence, and base verb semantics respectively. As we can see, the performance is abysmal. The features under-perform a random baseline of 50%, which is a more fitting baseline than a choose-first baseline of selecting the most common label which has an accuracy of 51.3%

<sup>4</sup>In practice we used row major form till the matrix was symmetricized for simplicity with the sparse matrix encoding we used, so the sparsification happened over rows

	Eval Set Accuracy	Testing Accuracy
i=2.0	0.516	0.484
i=2.25	0.501	0.500
i=2.5	0.518	0.484
i=2.75	0.506	0.484
i=3.0	0.500	0.493

Table 6.5: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using the sparsified semantics generated from PropBank frames’ role descriptions.

	Eval Set Accuracy	Testing Accuracy
i=2.0	0.524	0.475
i=2.25	0.520	0.477
i=2.5	0.525	0.482
i=2.75	0.526	0.478
i=3.0	0.514	0.490

Table 6.6: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using the sparsified semantics generated from co-occurrence in narrative chains.

since the means of representation we use removes ordering as a consideration (this is also desirable since ordering effects are obviously merely an artifact of the dataset here).

While these features are clearly useless on their own for Story Cloze, it is nonetheless important to see how they could be used to improve other feature sets, which would ultimately be their true purpose. To test that, we implemented a simple sentiment progression feature which scores the compound sentiment scores, as determined by the Vader sentiment analysis engine [Hutto and Gilbert, 2014], and produces a feature vector comprised of scores for each sentence. Similar features were found to be especially salient by many of the teams when competing the LSDSem2017 Shared Task, most notably in *mflor* [Flor and Somasundaran, 2017]. Using these sentiment sequence features on their own yields a training accuracy of 59.6% and a testing accuracy of 59.0%. Including the analogically driven features gives us the results shown in Tables 6.8, 6.9, and 6.10. In these we can see a stark contrast between the three semantic representations. Only by using PropBank frames’ role descriptions are we able to improve over the baseline of using sentiment scores without analogy features. In fact, in Table 6.8 all inflation parameters tested showed improvements

	Eval Set Accuracy	Testing Accuracy
i=2.0	0.523	0.499
i=2.25	0.491	0.505
i=2.5	0.513	0.495
i=2.75	0.512	0.498
i=3.0	0.511	0.493

Table 6.7: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using only the embeddings of the base verb.

	Testing Accuracy	% ER Over Sentiment
i=2.0	0.593	0.59
i=2.25	0.592	0.39
i=2.5	0.600	2.41
i=2.75	0.598	1.83
i=3.0	0.597	1.63

Table 6.8: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced with PropBank frames’ role descriptions joined with sentiment scores for each sentence. Included are the percentage error reductions over using the sentiment scores alone.

	Testing Accuracy	% ER Over Sentiment
i=2.0	0.577	-3.26
i=2.25	0.580	-2.48
i=2.5	0.578	-3.00
i=2.75	0.577	-3.20
i=3.0	0.580	-2.54

Table 6.9: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using the sparsified semantics generated from co-occurrence in narrative chains joined with sentiment scores for each sentence. Included are the percentage error reductions over using the sentiment scores alone.

over baseline, with a maximal score found at  $i = 2.5$  where the accuracy was 60% constituting a 2.41% error reduction over the sentiment scores alone. Viewing these results, it seems that the contribution of analogy using the PropBank frames’ role description semantics is a meaningful one, and indeed the results obtained are roughly in line with the 8<sup>th</sup> best performer in the LSDSem2017 shared task [Mostafazadeh et al., 2017] *Pranav.Goel* [Goel and Singh, 2017]. However, in absolute, the results obtained are not especially impressive, especially considering that higher performing models such as *msap* [Schwartz et al., 2017] and *cogcomp* [Peng et al., 2017] have accuracies close to 75%. Indeed, if we change the kernel in our SVM to a radial basis function (RBF), the performance of the sentiment baseline jumps up to 64%, but the improvements from our analogy features disappear<sup>5</sup>.

Understanding more about what utility there is to our features can help to address some of the questions about how they can be used, and what parts of them are useful. To that end, we also looked at how the features performed with the clustering features removed, focusing on the best performing semantic representation technique, namely PropBank Frames’ role description. When considered on their own, we found both an evaluation set and testing set accuracy of 49.9%, putting it roughly in line with the scores found when using clusters, but when considered in conjunction with the sentiment features we found an accuracy of

<sup>5</sup>In fact, all improvements from our feature set disappear when not using a linear kernel suggesting the features themselves have an especially meaningful linear relation which the kernel captures. The other obvious thought would be that the improvements from our feature are still there, but too minor to effect the improved baseline (suggesting that the features themselves are not ultimately salient). However, this does not hold up to scrutiny as when using a polynomial kernel, the performance of the sentiment baseline is not substantially improved over linear but the improvements from analogical features still vanish.

	Testing Accuracy	% ER Over Sentiment
$i=2.0$	0.588	-0.52
$i=2.25$	0.581	-1.96
$i=2.5$	0.582	-2.22
$i=2.75$	0.589	-0.39
$i=3.0$	0.586	-0.98

Table 6.10: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced with verb-role semantics made using only the embeddings of the base verb. The embeddings are joined with sentiment scores for each sentence to produce a full feature set for each potential story ending. Included are the percentage error reductions over using the sentiment scores alone.

59.7% with an error reduction of 1.67% over the sentiment-only baseline. This substantially under-performs the 2.41% error reduction found when including the cluster-based features (when using an inflation parameter of  $i = 2.5$ ). This ultimately means that analogy based narrative-clusters are an important part of the improvements we find from using analogical features for this task since what we fundamentally care about is the improvements our features can impart to other models, rather than their ability to stand on their own. Indeed, throughout most of our test cases, we found that training accuracy was largely contraindicative of testing accuracy when evaluating the features on their own, suggesting that there is some element of over-fitting. The total differences we found were minimal, and given the poor overall performance, smoothing to improve the performance is not a worthwhile endeavor.

And so, now that we know that, while the features do ‘work’ in some sense, the task is still better served by throwing them away to allow for better use of other, simpler, features, it becomes important to understand why the feature failed, and perhaps to hypothesize about what would be needed to make them functionally useful.

### 6.5.1 Follow-up Experimentation

To address the poor performance of the clustering features, here we’ll identify and explore several potential issues that may be the cause:

1. Semantics: Perhaps the semantic sparsification left too little semantic overlap, resulting in poor discovery of analogically similar narratives.
2. Analogy: Perhaps the measure of analogy we came up with was insufficient to properly capture real analogical relations.
3. Coverage: The number of potential analogies might be too small given the size of our dataset, and so we just couldn’t find a good match in the first place.



	Eval Set Accuracy	Testing Accuracy
i=2.0	0.522	0.507
i=2.25	0.525	0.494
i=2.5	0.520	0.484
i=2.75	0.501	0.500
i=3.0	0.516	0.485

Table 6.11: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using dense semantics generated from PropBank frames’ role descriptions.

4. Higher-Order Restrictions: Maybe successfully using measures of analogical similarity requires a higher order restriction such as from pragmatics, world-knowledge, or cognition to determine if a mapping is worthwhile to make at all.

We can now endeavor to rule out each of these in turn. The first is quite simple as we can replace our semantics with dense ones. The second requires that we use a different, more complete, model for analogy. Fortunately, the LISA model meets all our requirements, and we can look into results that it generates. We can address the third case by observing the results of the second. If LISA is able to find good analogical matches for most stories in our test set, we know that we have sufficient coverage from our training set. Ruling out higher-order restrictions is a bit more difficult, and without hand-coding every story pair which is prohibitive<sup>6</sup> we cannot be sure. However, by observing specific instances of LISA’s mapping results, we can get a good idea of what higher-order problems our method is susceptible to.

### Dense Semantics

Since semantic sparsification may be a cause for introducing error, we re-ran all of our analysis with the dense semantic embeddings which were used to generate the sparse encodings used above<sup>7</sup>. Table 6.11 shows the results obtained for the best performing semantic embedding, the PropBank based semantic encoding and Table 6.12 shows the results of the same embedding used in conjunction with sentiment features. While better than sparse semantic features alone (see Table 6.5), topping out at 50.7%, when combined with the sentiment scores as shown in Table 6.12, it under-performs the sparse features (see Table 6.8). Given that performance is still lacking, we can safely conclude that the sparse semantics are not the cause of the poor performance we found.

<sup>6</sup>The training set generates roughly 10 billion pairs for clustering, and the evaluation and test sets would require closer to 200 million pairs for evaluation

<sup>7</sup>N.B. The matrix for clustering was still sparsified, all we have done here is change the semantic representation to use dense vectors. By the time this reaches the transition matrix, each comparison, regardless of semantic space dimensionality is only a single number

	Testing Accuracy	% ER Over Sentiment
i=2.0	0.593	0.59
i=2.25	0.594	0.78
i=2.5	0.596	1.30
i=2.75	0.594	0.84
i=3.0	0.592	0.33

Table 6.12: Accuracy on the Story Cloze Task using inflation parameters from  $i = 2.0$  to  $i = 3.0$  with verb-role semantics produced using dense semantics generated from PropBank frames’ role descriptions. The embeddings are then joined with sentiment scores for each sentence to produce the final features. Included are the percentage error reductions over using the sentiment scores alone.

### LISA for Story Cloze

A more difficult prospect to check than semantic sparsity is whether our measure of analogical similarity fails to capture something important to the task which a more complete analogy would. Fortunately we have a means of measuring the analogical similarity between stories through full-blown analogical mapping as performed by LISA. Unfortunately we cannot run LISA between every pair stories to get measures for clustering, nor can we even run it between each training story and our evaluation and testing set stories since it is too computationally expensive. Instead, to make the best use of LISA, for each testing (or evaluation) story we find the five most semantically similar stories from the training set and include them with it into a single LISA simulation and allow LISA to discover mappings between the testing story and each of the training stories and vice-versa. To determine semantic similarity, we represent each story with two semantic embeddings, one for objects and the other for verb-roles. To get these single semantic embeddings, we take the average of the semantics of all objects or verb-roles. These two semantic vectors are then compared by cosine similarity against corresponding semantic vectors for all the testing stories with a final score constructed by taking a weighted average of the two cosine similarities. The weights were chosen as 75% verb-role and 25% object semantics to match LISA’s default predicate over object preference ratio of 3-to-1 which it uses in mapping discovery. Once these stories were selected, their LISAese representations were joined into a single LISA simulation with the testing (or evaluation) story and run such that LISA’s mapping was first driven by the testing story and it considered all propositions therein before deciding on any mappings (though not human realistic, this allows for the most accurate possible mapping). This mapping was reinforced further by allowing each training story to drive mapping where LISA updated its mapping confidences after each proposition, locking in the discovered mapping. The ultimate scores were produced by the averages and maximums of LISA’s outgoing and incoming mapping scores between the testing story and the training stories selected giving us four total features for each of the two possible story endings. These features were then fed into a linear kernel SVM for classification and, as before, we selected the story completion which SVM

assigned a higher probability to. We also explored using the LISA features in conjunction with the sentiment baseline. The resulting accuracies of the SVMs were 50.6% for LISA alone and 56.5% with the sentiment features for a decrease from the sentiment only baseline with an error reduction of  $-6.13\%$ . Given this poor performance, we can also safely rule out the analogical shortcuts we took as our problem.

### **Coverage**

It is quite possible that we do not have sufficient coverage from our training set, and that our method is failing because it simply cannot find any suitably analogous stories. If this were the case, we would expect to see a small maximum analogical similarity between most of the stories in our testing and evaluation sets with those from our training set. However, when looking at the results from our LISA features, we find an average maximum analogical similarity of roughly 66% which persists across both our testing and evaluation sets. For LISA, this is a rather high mapping score, and almost enough to trigger inference using the default thresholds. Indeed, roughly 54% of the stories **did** have sufficient similarity to trigger inference, implying that LISA thought they were analogous enough to warrant generating a schema. This constitutes a rather decent coverage, certainly enough that if this were our main problem, we would expect to see much better performance. Consequently, we can conclude that, while coverage is not a perfect ideal, it is certainly not the root cause of the awful performance we have encountered.

### **Higher-Order Restrictions**

The fact that LISA performed worse than our simpler analogical measure/clustering technique despite having a better and more complete analogy similarity score suggests that the important and useful stories which our large-scale approach made use of were not among those recalled by the object/verb-role semantic similarity method we used. And the fact that LISA still displayed a reasonable coverage suggests that there is a more fundamental problem at play. After looking more closely at individual examples of LISA simulations, we found that in many instances what LISA found that it considered a viable analogy was in fact a poor indicator for the task itself. The crux of the issue lies with the fact that an analogy with a good structural and semantic match does not necessarily mean a useful analogy. To better understand this difference, we can look at the following story:

Lisa wanted to adopt a pet. She was in love with both a puppy and a kitten but could only afford one. The puppy was more cuddly but the kitten was softer. As she picked it up, the kitten scratched her. In the end she bought the puppy.

By looking only at the predicates, objects, or events in this story, we would be likely to recall other stories of pet adoption, and in fact that is exactly what the method described above does. However, for the purposes of analogy the story can be more aptly described as a story about choice where pet adoption is merely the setting. Understanding that the story is about choice is quite difficult since there are no predicates in the story that have anything explicitly to do with choosing one option over another. To understand that wanting two things but only being able to get one requires a choice belongs to the domain of pragmatics and remains an open and very difficult problem. This discovery suggests that the issue of our method is centered around the necessity of a higher order restriction over the pragmatics of the stories under consideration for analogy.

## 6.6 Conclusion

The Story Cloze Task remains of great interest to the field, and as the task is improved, may become one of the first mainstream approachable tasks to require narrative comprehension. The model features we presented here simply do not stand up to the problem on their own. But we believe that, owing to their heavy dependence upon narrative structural alignment, they will prove to be beneficial to many state-of-the-art solutions to the task, as they were with a feature set of sentiment scores. This would allow them to potentially be used to improve performance in other models. However, while current solutions could potentially be improved with the addition of our features, since the task is still relatively new and there remains a great deal of room for improvement, we do not believe that it is a worthwhile endeavor to include our features given the likely small improvement and great computational cost associated with our method. As the task becomes more mature and improvements to accuracy become more stagnant, if other models do not make use of structural alignment and forms of analogical mapping, it may be worth revisiting our techniques to provide small bumps to a future state-of-the-art.

After a failure analysis of our feature set, we determined the most likely cause to be one which is deeply fundamental to narrative analogy as a whole. Namely, that finding analogies is not enough and instead we must also know **when** to find analogies. Without any means to determine which stories are pragmatically salient to one another, the utility of our features, be they direct analogical similarity or analogical clustering, is extremely limited for story-based tasks. We believe this will hold true regardless of the size of our training set as plenty of ‘good’ analogies can be found to pragmatically unrelated stories at very nearly the same rate as useful ones.

The limiting factor is not a failure or fault of the model or theories of analogy we used, nor is it a fault of the measure of analogical similarity we developed. Instead, it is a result of the basic premise of the task

which is that given two stories, we want to discover the best possible mapping between them. Unfortunately, many ‘good’ mappings are not useful ones. And until we can consistently identify relevant mappings, the use of analogy for open domain and broad-scope tasks such as for narrative comprehension, inference, or reasoning will likely remain difficult.

# Chapter 7

## Contributions

We set about to accomplish two main tasks:

1. Improve the LISA model of analogy
2. Analyze the usability of narrative analogy as a feature for story-based NLP tasks.

In tackling this we made several contributions split across the two tasks. The improvements we made to LISA fall into three main categories: algorithmic improvements, confidence measure, and automating the encoding of the model. In exploring narrative analogy as a feature we constructed a cheap measure of analogical similarity suitable for large scale applications, used it to cluster narratives, and ultimately analysed the measure's usefulness for the Story Cloze Task. Additionally, through failure analysis, we discovered what major problems open-domain use of narrative analogical features face and what future directions of research will be necessary before the features can see wide-spread use.

### 7.1 Improvements to LISA

The LISA model of analogy is one of the leading models of analogical reasoning and inference studied in the cognitive modeling world today. It has seen wide use both in and out of its field, and has had a large impact on the direction cognitive modeling of analogy as a whole. However, like all models, LISA has weaknesses. We focused mainly on addressing the following:

1. LISA's mapping algorithm was not robust to edge-cases, with issues potentially arising from ordering effects to weak semantic weights to poor mappings.
2. LISA's inferential capabilities made the model good at suggesting what was possible, but the model had no means of determining what was probable (a common complaint levied against the model by authors of the model's competitors)
3. Like every open-domain model of analogy, LISA's performance was entirely dependent upon the modeler's choice of encoding, making it difficult to impossible to determine what effects were the result of

theoretical claims made by the model versus effects wrought only by the modeler’s choice of representation.

### **7.1.1 LISA Algorithm**

The changes we made to LISA’s algorithm can be mostly grouped up into three main categories: Multi-Way Mapping, Mapping Algorithm Changes, and Activation Function Changes.

#### **Multi-Way Mapping**

To make LISA more accessible and useful for modeling analogical processes outside of cognitive science, we extended the model to allow it to find analogies between an arbitrary number of analogs simultaneously. These changes required an overhaul of the way that LISA initialized its mapping structure since multi-way mapping allowed for more locally optimal mappings which, when considered pair-wise across multiple stories, became inconsistent.

Ultimately, after that change, and some additional tweaks, the model was able to accurately handle mapping between multiple stories, successfully passing stress tests of mapping nearly a dozen stories at once.

#### **Mapping Algorithm Changes**

The largest and most fundamental changes made focused on the core of LISA, its mapping algorithm. We extended consistency checks between potential mappings to include more complex inconsistency checks, allowing LISA to avoid inconsistent mappings sooner, and thus be less likely to end up trapped in sub-optimal states. We also fundamentally changed the way that mappings are discovered by including additional considerations of global consistency to the checks LISA performs, allowing the model to avoid falling into order-based sub-optimal states from which it often could not escape.

#### **Activation Function Changes**

Lastly, we updated the way that LISA activates its units, changing its semantic activation as well as input aggregation.

LISA was unable to handle small semantic weights, making many word embeddings unsuited for its semantic representation. To address this we altered LISA’s semantic activation algorithm to make it closer to cosine similarity, allowing the model to seamlessly handle small semantic weights and thereby allow the use of general purpose word embeddings.

The way that LISA's unit activation worked, outside semantics, meant that if no good mappings were found, depending upon the presence of minor semantic activation, units would slowly aggregate input and grow to reach low-grade activation. Since LISA uses co-activation to determine mapping fitness, this low-grade activation, often across multiple competing units, could result in messy and poor mappings being discovered. To combat this, we put in place new checks to ensure sufficient levels of activation were reached sufficiently quickly before allowing continued aggregation, thereby making the model avoid these poor mappings.

Together, these three categories of algorithmic changes allow LISA to behave properly in the face of edge cases which previously stymied proper analogy production. By expanding the model's coverage to handle messier scenarios, we open the model up for broader use, especially in cases where a human modeler is not present to sanitize the input structure or order of activation.

### **7.1.2 Confidence Measure**

LISA has always been used to generate schemas by extending analogies, indeed "schemas" is responsible for the 'S' in LISA. However, the schemas which LISA produces are not statistical in nature, and as such make no claims as to the likelihood of any given event. This omission has long been a point of contention for those who favor its competitors. To address these concerns, we developed an algorithm to calculate a belief, called Inductive Confidence, in the truth or falsehood of any proposition represented in LISA. Using Inductive Confidence we showed that belief in propositions can be updated based upon LISA's discovered mappings, and that over the course of several mappings across multiple analogs LISA can develop a statistical representation for its belief in the likelihood of a given event in any story, most notably useful for schematization.

### **7.1.3 Automation of Encoding**

In all but a couple niche models of analogy focused on small domains the modeler is an inextricable part of the model. This raises several issues, most notably that whatever results the model produces, and thereby whatever claims the model supports, are as much a result of the modeler's choice of representation as they are of the model itself. To combat this, we presented an automated way of producing LISA simulations from natural text, thereby removing the need for a modeler's input into representation for analogy. This allows analysis of the claims of the model independent of the modeler's biases.

Automation also opens up avenues for the application of LISA to large scale datasets which would have been prohibitively time-consuming to hand-code. This makes LISA an option for use in a spectrum of AI applications.



Unfortunately, the more complex the syntax of the text is, the less likely our method is to produce a proper LISAese representation since every parser, labeler, and tagger must all produce the correct output to get the correct LISA encoding. This fragility to the encoder limits the scope to which LISA can be applied. Future work should expand the coverage of LISA encoding by building more a robust framework between the NLP tools it makes use of, hopefully removing the need for single-point failure.

## **7.2 Narrative Analogy as a Feature**

Analysis of narrative structure has proven to be a fruitful domain for research with a host of studies focused on it coming out in recent years. More specifically, there have been several approaches to story-based tasks which make heavy use of narrative analogy to great effect. However, each of these focuses on a more traditional use of story structure using either folktales or fables. Instead, here we explored the usability of narrative analogy for an open-domain story-based task, the Story Cloze Task. In doing so, we made two contributions: we presented a novel measure of analogical similarity and, as our main contribution, a failure analysis over the use of the feature set, allowing us to understand the limitations of narrative analogy for the kind of open-domain use that Story Cloze represents.

### **7.2.1 Analogical Similarity Measure**

Calculating the analogical similarity between narratives is either limited in its expressivity, as is the case with SME’s sub-graph-size measure, or is prohibitively computationally expensive, as is the case with LISA’s mapping score. To remedy this, we developed a measure of analogical similarity based off a combination of LISA’s recall and mapping functionality which was both computationally cheap and allowed for meaningful analysis of how well individual actors analogically aligned with one another. We used this similarity measure to cluster narratives in the hopes of approximating schematization, finding that the resulting clusters gave a minor improvement to performance. The features generated from the measure and clusters proved to be poor indicators for the Story Cloze Task on their own, but did show more promise when combined with a sentiment score, improving the accuracy over the sentiment score alone with an error reduction of nearly two and half percent.

During this process we also explored three potential semantic representations for verb-roles in analogy making. Ultimately we found that incorporating the knowledge base of PropBank into the semantics vastly outperformed our other two encodings, one an embedding of the verb itself, ignoring the differences between roles, the other a direct embedding of the verb-role by using co-reference chains as co-occurrence windows

for traditional word embedding techniques.

The fundamental contribution of this measure lies more with the failure analysis it allowed rather than an intrinsic benefit from the measure itself. That said, future work with narrative analogy should make note of the semantic representational differences we observed and include PropBank Frames’ role descriptions in their verb-role semantics.

### **7.2.2 Discovered Problems With Narrative Analogy**

As narrative structure is becoming a more prevalent topic in NLP, it is important to understand what potential analogy has for interpreting and understanding narrative. Narrative analogy has already been shown to be useful for limited domains, but when expanding the scope of stories over which we want to draw analogies, new problems arise. In the process of failure analysis on the analogical features we used for the Story Cloze Task, we found that in many cases what the analogy modeling community would term ‘good’ analogies were nonetheless useless for the Story Cloze Task. That is, just because an analogy fits together well does not make it a useful one. Ultimately, knowing how to make analogies is not enough, we must also know *when* to make them. This distinction is one of pragmatic relevance and salience which requires a deeper understanding of what the stories are actually about in order to solve. Perhaps development of an analogically driven story ontology into which stories could be classified by high-level type to allow for meaningful analogies to be drawn would not require full-blown pragmatic understanding, but until we can determine whether an analogy *should* be made, it does not appear worthwhile for general open-domain story-based tasks to tell whether an analogy *could* be made.

# Appendix A

## LISA

Here we will cover the actual equations used in LISA during mapping discovery

### A.1 SP Unit Firing Order

To determine the firing order of SP units, LISA uses a compound process by which activation of each SP unit is determined by their sensitivity to inhibition and an internal inhibitor which are dynamically updated during the run of a phase-set. The idea behind the sensitivity to inhibition is to help ensure that every SP unit gets a chance to fire before any individual SP fires twice. In practice, LISA has a functionality to remove these calculations and simply force each SP unit to fire in order, however for modeling human performance, this removes the element of a working memory constraint (can think about an arbitrary number of role-filler bindings – which can be humanly impossible).

The way that the equations are set is to ensure that as one SP unit fires, its sensitivity to inhibition increases during its run till it grows too large and the unit is inhibited, at which time the next SP unit can begin firing as any unit not firing currently has their sensitivity slowly decreased. After LISA has moved on to another SP unit, the original one which finished firing begins to recover its sensitivity to inhibition, eventually lowering enough to allow the unit to become active once again. Worth noting is that other SP units, which have not yet fired at all by the time the sensitivity of the first firing SP unit lowers enough to allow activation, will have an even lower sensitivity to inhibition (as theirs has been falling the entire time).

$$\Delta_{STI} = \delta_{STI} \cdot (1 + \delta_{noise} \cdot \pi|_{-1}^1) \quad (\text{A.1})$$

Here the change to the sensitivity to inhibition is decreased ( $\delta_{STI}$  is negative) by a fixed amount set by a model parameter plus or minus some random noise ( $\pi|_{-1}^1$ ) whose size is determined by another model parameter ( $\delta_{noise}$ ).

This equation is then used in conjunction with an inhibitor updating algorithm to switch between SP units during a phase-set. The inhibitor algorithm is such that once the activation of a given SP unit is above

some model parameter determined threshold (the model has decided that the SP unit is currently getting its ‘turn’) the inhibitor will rise, otherwise it will decrease since its not currently active and should prepare to potentially become active in the future. During both the growth and decay of the inhibitor for each SP unit there are two distinct phases, a fast growth/decay and a slow growth/decay. While the inhibitor is small it grows slowly, but once it passes a fixed threshold of 0.1, it grows much faster till it hits 1.0. During this rapid inhibitor growth phase LISA will also set the SP unit’s sensitivity to inhibition to a maximum value since this is the period during which LISA is switching which SP unit gets to be active. Similarly, during decay, while the inhibitor is quite large, above 0.9, it will experience a slow decay rate, but speed up considerably once it falls below that threshold. With this, we can now see the formulas LISA uses in SP unit inhibition updating.

$$SP\_inhibition = 2 \cdot act_{max} \cdot SP\_STI \cdot \lambda_{inhib} + SP\_inhibitor \cdot \Gamma_{inhibitor} \quad (A.2)$$

The actual inhibition of each SP unit is calculated as two times the product of the activation of the most-active other SP unit, its sensitivity to inhibition, and an inhibition rate plus the product of the inhibitor and a model parameter.

## A.2 Unit Activation

There are specific instances where the general unit activation equation is different from the norm, but at its fundamental, unit activation has an asymptotic growth and decay rate.

$$\Delta_{act}(input) = \lambda_{growth} \cdot input \cdot (1 - act) - \lambda_{decay} \cdot act \quad (A.3)$$

The thrust of this equation is that the change to a unit’s activation given some input at a fixed time-step can be broken apart into two pieces dictated by  $\lambda_{growth}$  and  $\lambda_{decay}$  the change-rate for growth and decay respectively.

The effect of the asymptotic behavior is that as the activation grows larger, the growth decreases while the decay increases, and vice versa with a larger growth potential and negligible decay at low activation.

# References

- [Azad et al., 2018] Azad, A., Pavlopoulos, G. A., Ouzounis, C. A., Kyrpides, N. C., and Buluç, A. (2018). HipMCL: A high-performance parallel implementation of the Markov clustering algorithm for large-scale networks. *Nucleic Acids Research*, 46(6):e33–e33.
- [Baker et al., 1998] Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 17th International Conference on Computational Linguistics*, volume 1, pages 86–90. Association for Computational Linguistics (ACL).
- [Bavelas, 1950] Bavelas, A. (1950). Communication patterns in task-oriented groups. *Journal of the Acoustical Society of America*, 22:725–730.
- [Baydin et al., 2015] Baydin, A. G., de Mántaras, R. L., and Ontañón, S. (2015). A semantic network-based evolutionary algorithm for computational creativity. *Evolutionary Intelligence*, 8(1):3–21.
- [Bex et al., 2007] Bex, F. J., Prakken, H., and Verheij, B. (2007). Formalising argumentative story-based analysis of evidence. In *Proceedings of the 11th International Conference on Artificial Intelligence and Law*, pages 1–10. Association of Computing Machinery (ACM).
- [Brody, 2007] Brody, S. (2007). Clustering clauses for high-level relation detection: An information-theoretic approach. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics*, pages 448–455.
- [Caporale and Dan, 2008] Caporale, N. and Dan, Y. (2008). Spike timing-dependent plasticity: A Hebbian learning rule. *Annual Review of Neuroscience*, 31:25–46.
- [Carlson, 2019] Carlson, M. (2019). 600 free ESL short stories from New York city. <https://eslyes.com/nyc/contents.htm>.
- [Catrambone and Holyoak, 1989] Catrambone, R. and Holyoak, K. J. (1989). Overcoming contextual limitations on problem-solving transfer. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(6):1147.
- [Chambers, 2013] Chambers, N. (2013). Event schema induction with a probabilistic entity-driven model. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807.
- [Chambers and Jurafsky, 2008] Chambers, N. and Jurafsky, D. (2008). Unsupervised learning of narrative event chains. In *Association for Computational Linguistics (ACL)*, pages 789–797, Hawaii, USA.
- [Chambers and Jurafsky, 2009] Chambers, N. and Jurafsky, D. (2009). Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 2, pages 602–610. Association for Computational Linguistics.
- [Chambers, 2011] Chambers, N. W. (2011). *Inducing Event Schemas and Their Participants from Unlabeled Text*. PhD thesis, Stanford University.

- [Chatman, 1980] Chatman, S. B. (1980). *Story and discourse: Narrative structure in fiction and film*. Cornell University Press, Ithica, NY.
- [Chaturvedi et al., 2017] Chaturvedi, S., Peng, H., and Roth, D. (2017). Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614.
- [Chen and Ji, 2010] Chen, Z. and Ji, H. (2010). Graph-based clustering for computational linguistics: A survey. In *Proceedings of the 2010 Workshop on Graph-Based Methods for Natural Language Processing*, pages 1–9. Association for Computational Linguistics.
- [Cheng, 1995] Cheng, Y. (1995). Mean shift, mode seeking, and clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790–799.
- [Cheung et al., 2013] Cheung, J. C. K., Poon, H., and Vanderwende, L. (2013). Probabilistic frame induction. *arXiv preprint arXiv:1302.4813*.
- [Clement and Gentner, 1991] Clement, C. A. and Gentner, D. (1991). Systematicity as a selection constraint in analogical mapping. *Cognitive Science*, 15(1):89–132.
- [Dehghani et al., 2008] Dehghani, M., Tomai, E., Forbus, K., Iliev, R., and Klenk, M. (2008). Moraldm: A computational model of moral decision-making. In *Proceedings of the 30th Annual Conference of the Cognitive Science Society (CogSci)*, Washington, DC.
- [Demski et al., 2014] Demska, A., Ustun, V., Rosenbloom, P., and Kommers, C. (2014). Outperforming Word2Vec on analogy tasks with random projections. *arXiv preprint arXiv:1412.6616*.
- [Dorow et al., 2004] Dorow, B., Widdows, D., Ling, K., Eckmann, J.-P., Sergi, D., and Moses, E. (2004). Using curvature and Markov clustering in graphs for lexical acquisition and word sense discrimination. *arXiv preprint cond-mat/0403693*.
- [Doumas et al., 2008] Doumas, L. A., Hummel, J. E., and Sandhofer, C. M. (2008). A theory of the discovery and predication of relational concepts. *Psychological Review*, 115(1):1.
- [Dunn et al., 2008] Dunn, C. W., Hejnol, A., Matus, D. Q., Pang, K., Browne, W. E., Smith, S. A., Seaver, E., Rouse, G. W., Obst, M., Edgecombe, G. D., et al. (2008). Broad phylogenomic sampling improves resolution of the animal tree of life. *Nature*, 452(7188):745.
- [Eisner, 2003] Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics (ACL)*, volume 2, pages 205–208.
- [Elson, 2012] Elson, D. K. (2012). *Modeling Narrative Discourse*. PhD thesis, Columbia University.
- [Enright et al., 2002] Enright, A. J., Van Dongen, S., and Ouzounis, C. A. (2002). An efficient algorithm for large-scale detection of protein families. *Nucleic Acids Research*, 30(7):1575–1584.
- [Falkenhainer et al., 1989] Falkenhainer, B., Forbus, K. D., and Gentner, D. (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence*, 41(1):1–63.
- [Faruqui et al., 2015] Faruqui, M., Tsvetkov, Y., Yogatama, D., Dyer, C., and Smith, N. A. (2015). Sparse overcomplete word vector representations. *Computing Research Repository (CoRR)*, abs/1506.02004.
- [Fechner et al., 1966] Fechner, G. T., Howes, D. H., and Boring, E. G. (1966). *Elements of Psychophysics*, volume 1. Holt, Rinehart and Winston New York.
- [Finlayson, 2012] Finlayson, M. M. A. (2012). *Learning Narrative Structure from Annotated Folktales*. PhD thesis, Massachusetts Institute of Technology.

- [Flor and Somasundaran, 2017] Flor, M. and Somasundaran, S. (2017). Sentiment analysis and lexical cohesion for the Story Cloze task. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 62–67. Association for Computational Linguistics (ACL).
- [Forbus et al., 1994] Forbus, K., Ferguson, R., and Gentner, D. (1994). Incremental structure-mapping. In *Proceedings of the 16th Annual Conference of the Cognitive Science Society (CogSci)*, pages 313–318.
- [Forbus et al., 1995] Forbus, K. D., Gentner, D., and Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. *Cognitive Science*, 19(2):141–205.
- [Friedman and Forbus, 2009] Friedman, S. and Forbus, K. D. (2009). Learning naive physics models and misconceptions. In *Proceedings of the 31st Annual Conference of the Cognitive Science Society (CogSci)*, pages 2505–2510.
- [Garey and Johnson, 2002] Garey, M. R. and Johnson, D. S. (2002). *Computers and Intractability*, volume 29. WH Freeman, New York, NY.
- [Gentner, 1983] Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science*, 7(2):155–170.
- [Gentner, 1989] Gentner, D. (1989). The mechanisms of analogical learning. In *Similarity and Analogical Reasoning*, pages 199–241. Cambridge University Press.
- [Gentner, 2010] Gentner, D. (2010). Bootstrapping the mind: Analogical processes and symbol systems. *Cognitive Science*, 34(5):752–775.
- [Gentner and Forbus, 2011] Gentner, D. and Forbus, K. D. (2011). Computational models of analogy. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(3):266–276.
- [Gentner et al., 2001] Gentner, D., Holyoak, K. J., and Kokinov, B. N. (2001). *The analogical mind: Perspectives from cognitive science*. MIT Press, Cambridge, MA.
- [Gentner and Kurtz, 2006] Gentner, D. and Kurtz, K. J. (2006). Relations, objects, and the composition of analogies. *Cognitive Science*, 30(4):609–642.
- [Gentner and Markman, 1997] Gentner, D. and Markman, A. B. (1997). Structure mapping in analogy and similarity. *American Psychologist*, 52(1):45.
- [Gentner and Markman, 2006] Gentner, D. and Markman, A. B. (2006). Defining structural similarity. *The Journal of Cognitive Science*, 6(1):1–20.
- [Gentner et al., 1993] Gentner, D., Rattermann, M. J., and Forbus, K. D. (1993). The roles of similarity in transfer: Separating retrievability from inferential soundness. *Cognitive Psychology*, 25(4):524–575.
- [Gick and Holyoak, 1983] Gick, M. L. and Holyoak, K. J. (1983). Schema induction and analogical transfer. *Cognitive Psychology*, 15(1):1–38.
- [Goel and Singh, 2017] Goel, P. and Singh, A. K. (2017). IIT (BHU): System description for LSDSem 2017 shared task. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 81–86. Association for Computational Linguistics (ACL).
- [Goyal et al., 2010] Goyal, A., Riloff, E., and Daumé III, H. (2010). Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86. Association for Computational Linguistics.
- [Green, 2012] Green, G. M. (2012). *Pragmatics and natural language understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ.
- [Green and Dorr, 2005] Green, R. and Dorr, B. (2005). Frame semantic enhancement of lexical-semantic resources. In *Proceedings of the ACL-SIGLEX Workshop on Deep Lexical Acquisition*, pages 57–66.

- [Hebb, 1949] Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons.
- [Hebb, 2005] Hebb, D. O. (2005). *The organization of behavior: A neuropsychological theory*. Psychology Press.
- [Hinrichs and Forbus, 2011] Hinrichs, T. and Forbus, K. D. (2011). Transfer learning through analogy in games. *AI Magazine*, 32(1):70–70.
- [Hiroshi et al., 2004] Hiroshi, K., Tetsuya, N., and Hideo, W. (2004). Deeper sentiment analysis using machine translation technology. In *Proceedings of the 20th International Conference on Computational Linguistics*, page 494. Association for Computational Linguistics.
- [Hofstadter, 2001] Hofstadter, D. R. (2001). Analogy as the core of cognition. *The Analogical Mind: Perspectives from Cognitive Science*, pages 499–538.
- [Hofstadter et al., 1994] Hofstadter, D. R., Mitchell, M., et al. (1994). The Copycat project: A model of mental fluidity and analogy-making. *Advances in Connectionist and Neural Computation Theory*, 2(31-112):29–30.
- [Holyoak et al., 1996] Holyoak, K. J., Holyoak, K. J., and Thagard, P. (1996). *Mental leaps: Analogy in creative thought*. MIT Press, Cambridge, MA.
- [Holyoak and Koh, 1987] Holyoak, K. J. and Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition*, 15(4):332–340.
- [Holyoak and Morrison, 2005] Holyoak, K. J. and Morrison, R. G. (2005). *The Cambridge Handbook of Thinking and Reasoning*. Cambridge University Press, Cambridge, MA.
- [Holyoak and Thagard, 1989] Holyoak, K. J. and Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science*, 13(3):295–355.
- [Huang et al., 2013] Huang, P.-S., He, X., Gao, J., Deng, L., Acero, A., and Heck, L. (2013). Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, pages 2333–2338. ACM.
- [Hummel and Holyoak, 1997] Hummel, J. E. and Holyoak, K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*, 104(3):427.
- [Hummel and Holyoak, 2003] Hummel, J. E. and Holyoak, K. J. (2003). A symbolic-connectionist theory of relational inference and generalization. *Psychological Review*, 110(2):220.
- [Hummel and Holyoak, 2005] Hummel, J. E. and Holyoak, K. J. (2005). Relational reasoning in a neurally plausible cognitive architecture: An overview of the LISA project. *Current Directions in Psychological Science*, 14(3):153–157.
- [Hummel et al., 2004] Hummel, J. E., Holyoak, K. J., Green, C., Dumas, L. A., Devnich, D., Kittur, A., and Kalar, D. J. (2004). A solution to the binding problem for compositional connectionism. In *Compositional Connectionism in Cognitive Science: Papers from the AAAI Fall Symposium*, ed. SD Levy & R. Gayler, pages 31–34.
- [Hummel and Landy, 2009] Hummel, J. E. and Landy, D. H. (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*, pages 211–221. New Bulgarian University.
- [Hummel et al., 2008] Hummel, J. E., Landy, D. H., and Devnich, D. (2008). Toward a process model of explanation with implications for the type-token problem. In *AAAI Fall Symposium: Naturally-Inspired Artificial Intelligence*, pages 79–86.



- [Hummel et al., 2014] Hummel, J. E., Licato, J., and Bringsjord, S. (2014). Analogy, explanation, and proof. *Frontiers in Human Neuroscience*, 8.
- [Hutto and Gilbert, 2014] Hutto, C. J. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International AAI Conference on Weblogs and Social Media*. Association for the Advancement of Artificial Intelligence (AAAI).
- [Jans et al., 2012] Jans, B., Bethard, S., Vulić, I., and Moens, M. F. (2012). Skip n-grams and ranking functions for predicting script events. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 336–344. Association for Computational Linguistics.
- [Jonker and Volgenant, 1987] Jonker, R. and Volgenant, A. (1987). A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340.
- [Kann, 1992] Kann, V. (1992). *On the Approximability of NP-Complete Optimization Problems*. PhD thesis, Royal Institute of Technology Stockholm.
- [Karlsson et al., 2013] Karlsson, F. H., Tremaroli, V., Nookaew, I., Bergström, G., Behre, C. J., Fagerberg, B., Nielsen, J., and Bäckhed, F. (2013). Gut metagenome in european women with normal, impaired and diabetic glucose control. *Nature*, 498(7452):99.
- [Keane et al., 1994] Keane, M. T., Ledgeway, T., and Duff, S. (1994). Constraints on analogical mapping: A comparison of three models. *Cognitive Science*, 18(3):387–438.
- [Kingsbury and Palmer, 2002] Kingsbury, P. and Palmer, M. (2002). From treebank to propbank. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC2002)*, Las Palmas, Spain.
- [Kiros et al., 2015] Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S. (2015). Skip-thought vectors. In *Advances in Neural Information Processing Systems*, pages 3294–3302.
- [Klenk and Forbus, 2009] Klenk, M. and Forbus, K. (2009). Domain transfer via cross-domain analogy. *Cognitive Systems Research*, 10(3):240–250.
- [Kokinov and Petrov, 2001] Kokinov, B. and Petrov, A. (2001). Integrating memory and reasoning in analogy-making: The AMBR model. *The Analogical Mind: Perspectives from Cognitive Science*, pages 59–124.
- [Kokinov, 1988] Kokinov, B. N. (1988). Associative memory-based reasoning: How to represent and retrieve cases. In *Artificial Intelligence III: Methodology, Systems, Applications*, pages 51–58.
- [Kuehne et al., 2000] Kuehne, S., Forbus, K., Gentner, D., and Quinn, B. (2000). SEQL: Category learning as progressive abstraction using structure mapping. In *Proceedings of the 22nd Annual Meeting of the Cognitive Science Society*, volume 4, pages 770–775, Philadelphia, PA.
- [Kuhn, 1955] Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97.
- [Landau, 1984] Landau, M. (1984). Human evolution as narrative: Have hero myths and folktales influenced our interpretations of the evolutionary past? *American Scientist*, 72(3):262–268.
- [Larkey and Love, 2003] Larkey, L. B. and Love, B. C. (2003). CAB: Connectionist analogy builder. *Cognitive Science*, 27(5):781–794.
- [Lascarides and Asher, 2008] Lascarides, A. and Asher, N. (2008). Segmented discourse representation theory: Dynamic semantics with discourse structure. In *Computing Meaning*, pages 87–124. Springer.

- [Lee, 2019] Lee, R. C. (2019). ESL: English as a second language - free English learning resources. <https://www.rong-chang.com/>.
- [Li et al., 2003] Li, L., Stoeckert, C. J., and Roos, D. S. (2003). OrthoMCL: identification of ortholog groups for eukaryotic genomes. *Genome Research*, 13(9):2178–2189.
- [Lloyd, 1982] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.
- [Manning et al., 2014] Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60.
- [Manurung, 2004] Manurung, H. (2004). *An Evolutionary Algorithm Approach to Poetry Generation*. PhD thesis, University of Edinburgh. College of Science and Engineering. School of Informatics.
- [Markman and Gentner, 2000] Markman, A. B. and Gentner, D. (2000). Structure mapping in the comparison process. *American Journal of Psychology*, 113(4):501–538.
- [Marshall, 1995] Marshall, J. A. (1995). Adaptive perceptual pattern recognition by self-organizing neural networks: Context, uncertainty, multiplicity, and scale. *Neural Networks*, 8(3):335–362.
- [McGregor et al., 2016] McGregor, S., Purver, M., and Wiggins, G. (2016). Words, concepts, and the geometry of analogy. *arXiv preprint arXiv:1608.01403*.
- [Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.
- [Mikolov et al., 2013b] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119.
- [Mikolov et al., 2013c] Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751.
- [Miller, 1995] Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the Association of Computing Machinery (ACM)*, 38(11):39–41.
- [Mitchell, 1993] Mitchell, M. (1993). *Analogy-making as perception: A computer model*. MIT Press, Cambridge, MA.
- [Mooney and DeJong, 1985] Mooney, R. J. and DeJong, G. (1985). Learning schemata for natural language processing. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, pages 681–687.
- [Morrison et al., 2004] Morrison, R. G., Krawczyk, D. C., Holyoak, K. J., Hummel, J. E., Chow, T. W., Miller, B. L., and Knowlton, B. J. (2004). A neurocomputational model of analogical reasoning and its breakdown in frontotemporal lobar degeneration. *Journal of Cognitive Neuroscience*, 16(2):260–271.
- [Mostafazadeh et al., 2016a] Mostafazadeh, N., Chambers, N., He, X., Parikh, D., Batra, D., Vanderwende, L., Kohli, P., and Allen, J. (2016a). A corpus and cloze evaluation for deeper understanding of common-sense stories. *Proceedings of NAACL HLT, San Diego, California, June. Association for Computational Linguistics*.
- [Mostafazadeh et al., 2016b] Mostafazadeh, N., Grealish, A., Chambers, N., Allen, J., and Vanderwende, L. (2016b). CaTeRS: Causal and temporal relation scheme for semantic annotation of event structures. In *Proceedings of the 4th Workshop on Events: Definition, Detection, Coreference, and Representation*, pages 51–61, San Diego, CA. Association for Computational Linguistics (ACL).

- [Mostafazadeh et al., 2017] Mostafazadeh, N., Roth, M., Louis, A., Chambers, N., and Allen, J. (2017). LS-DSem 2017 shared task: The Story Cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51. Association for Computational Linguistics (ACL).
- [Mostafazadeh et al., 2016c] Mostafazadeh, N., Vanderwende, L., Yih, W.-t., Kohli, P., and Allen, J. (2016c). Story Cloze evaluator: Vector space representation evaluation by predicting what happens next. In *Proceedings of the 1st Workshop on Evaluating Vector Space Representations for NLP*, pages 24–29, Berlin, Germany.
- [Ng et al., 2002] Ng, A. Y., Jordan, M. I., and Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856.
- [Orr et al., 2014] Orr, J. W., Tadepalli, P., Doppa, J. R., Fern, X., and Dietterich, T. G. (2014). Learning scripts as hidden markov models. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.
- [Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.
- [Peng et al., 2017] Peng, H., Chaturvedi, S., and Roth, D. (2017). A joint model for semantic sequences: Frames, entities, sentiments. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 173–183.
- [Penn et al., 2008] Penn, D. C., Holyoak, K. J., and Povinelli, D. J. (2008). Darwin’s mistake: Explaining the discontinuity between human and nonhuman minds. *Behavioral and Brain Sciences*, 31(02):109–130.
- [Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.
- [Pichotta and Mooney, 2014] Pichotta, K. and Mooney, R. J. (2014). Statistical script learning with multi-argument events. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, volume 14, pages 220–229.
- [Pichotta and Mooney, 2016] Pichotta, K. and Mooney, R. J. (2016). Statistical script learning with recurrent neural networks. In *Proceedings of EMNLP 2016 Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16, Austin, TX.
- [Propp, 2010] Propp, V. (2010). *Morphology of the Folktale*, volume 9. University of Texas Press.
- [Punyakanok et al., 2008] Punyakanok, V., Roth, D., and Yih, W. (2008). The importance of syntactic parsing and inference in semantic role labeling. *Association for Computational Linguistics(ACL)*, 34(2).
- [Qin et al., 2010] Qin, J., Li, R., Raes, J., Arumugam, M., Burgdorf, K. S., Manichanh, C., Nielsen, T., Pons, N., Levenez, F., Yamada, T., et al. (2010). A human gut microbial gene catalogue established by metagenomic sequencing. *Nature*, 464(7285):59.
- [Rapaport et al., 1989] Rapaport, W. J., Segal, E. M., Shapiro, S. C., Zubin, D. A., Bruder, G. A., Duchan, J. F., Almeida, M. J., Daniels, J. H., Galbraith, M. M., Wiebe, J. M., et al. (1989). *Deictic centers and the cognitive structure of narrative comprehension*. State University of New York (Buffalo). Department of Computer Science.
- [Reiter, 2014] Reiter, N. (2014). *Discovering Structural Similarities in Narrative Texts Using Event Alignment Algorithms*. PhD thesis, Heidelberg University.
- [Ross, 1989] Ross, B. H. (1989). Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(3):456.

- [Rudinger and Van Durme, 2014] Rudinger, R. and Van Durme, B. (2014). Is the stanford dependency representation semantic? In *Proceedings of the 2nd Workshop on EVENTS: Definition, Detection, Coreference, and Representation*, pages 54–58, Baltimore, MD. Association for Computational Linguistics (ACL).
- [Schaeffer, 2007] Schaeffer, S. E. (2007). Graph clustering. *Computer Science Review*, 1(1):27–64.
- [Schank and Abelson, 2008] Schank, R. C. and Abelson, R. P. (2008). *Scripts, Plans, Goals and Understanding: An Inquiry into Human Knowledge Structures*. Psychology Press.
- [Schwartz et al., 2017] Schwartz, R., Sap, M., Konstas, I., Zilles, L., Choi, Y., and Smith, N. A. (2017). Story Cloze task: UW NLP system. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 52–55. Association for Computational Linguistics (ACL).
- [Sharma et al., 2018] Sharma, R., Allen, J., Bakhshandeh, O., and Mostafazadeh, N. (2018). Tackling the story ending biases in the Story Cloze test. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 752–757.
- [Skorstad et al., 1988] Skorstad, J., Gentner, D., and Medin, D. (1988). Abstraction processes during concept learning: A structural view. In *Proceedings of the 10th Annual Conference of the Cognitive Science Society (CogSci)*, pages 419–425.
- [Spellman and Holyoak, 1992] Spellman, B. A. and Holyoak, K. J. (1992). If Saddam is Hitler then who is George Bush? analogical mapping between systems of social roles. *Journal of Personality and Social Psychology*, 62(6):913.
- [Tenkasi and Boland, 1993] Tenkasi, R. V. and Boland, R. J. (1993). Locating meaning making in organizational learning: The narrative basis of cognition. *Research in Organizational Change and Development*, 7:77–103.
- [Thagard et al., 1990] Thagard, P., Holyoak, K. J., Nelson, G., and Gochfeld, D. (1990). Analog retrieval by constraint satisfaction. *Artificial Intelligence*, 46(3):259–310.
- [Tinker et al., 2005] Tinker, P., Fox, J., Green, C., Rome, D., Casey, K., and Furmanski, C. (2005). Analogical and case-based reasoning for predicting satellite task schedulability. In *Proceedings of the 2005 International Conference on Case-Based Reasoning*, pages 566–578, Chicago, IL. Springer.
- [Tomai and Forbus, 2009] Tomai, E. and Forbus, K. D. (2009). EA NLU: Practical language understanding for cognitive modeling. In *Proceedings of the 22nd International Florida Artificial Intelligence Research Society Conference*, Sanibel Island, FL.
- [Tversky, 1977] Tversky, A. (1977). Features of similarity. *Psychological Review*, 84(4):327.
- [Van Dongen, 2000] Van Dongen, S. M. (2000). *Graph Glustering by Flow Simulation*. PhD thesis, Utrecht University.
- [Vendetti et al., 2014] Vendetti, M. S., Wu, A., and Holyoak, K. J. (2014). Far-out thinking: Generating solutions to distant analogies promotes relational thinking. *Psychological Science*, 25(4):928–933.
- [Viskontas et al., 2004] Viskontas, I. V., Morrison, R. G., Holyoak, K. J., Hummel, J. E., and Knowlton, B. J. (2004). Relational integration, inhibition, and analogical reasoning in older adults. *Psychology and Aging*, 19(4):581.
- [Wang and Yang, 2011] Wang, H. and Yang, Q. (2011). Transfer learning by structural analogy. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. AAAI Press.
- [Wilner and Hummel, 2017] Wilner, S. and Hummel, J. (2017). Automating the encoding for the LISA model of analogy from raw text. *Modern Artificial Intelligence and Cognitive Science (MAICS)*, pages 115–122.

- [Winston, 1978] Winston, P. H. (1978). Learning by creatifying transfer frames. *Artificial Intelligence*, 10(2):147–172.
- [Xu and Wunsch, 2005] Xu, R. and Wunsch, D. C. (2005). Survey of clustering algorithms. *IEEE Transactions on Neural Network*, 16(3):645–678.
- [Zelenko et al., 2003] Zelenko, D., Aone, C., and Richardella, A. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3(Feb):1083–1106.
- [Zhang et al., 2016] Zhang, S., Rudinger, R., Duh, K., and Van Durme, B. (2016). Ordinal common-sense inference. *arXiv preprint arXiv:1611.00601*.
- [Zhu and Ontanón, 2010] Zhu, J. and Ontanón, S. (2010). Towards analogy-based story generation. In *Proceedings of the International Conference on Computational Creativity (ICCC)*, pages 75–84, Lisbon, Portugal.