BOSTON UNIVERSITY

GRADUATE SCHOOL OF ARTS & SCIENCES

Ph.D. Final Dissertation

DISAMBIGUATING NATURAL LANGUAGE VIA ALIGNING MEANINGFUL DESCRIPTIONS

by

YIDA XIN

B.S. in Mathematics, Tulane University, 2017
M.S. in Computer Science, Boston University, 2020

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2022

Approved by

First Reader

Sang Peter Chin, PhD
Research Professor of Computer Science

Second Reader

Steve Homer, PhD
Professor of Computer Science

Third Reader

Derry Wijaya, PhD
Assistant Professor of Computer Science

Fourth Reader

Henry Lieberman, PhD
Research Scientist of Computer Science and Artificial Intelligence Laboratory

Acknowledgments

My advisers, Professor Peter Chin and Henry Lieberman, have been supporting all my research endeavors in a never-ending fashion. Professors Steve Homer and Derry Wijaya have graciously provided their feedback on my thesis and attended my thesis defense as committee members.

Professors Jonathan Appavoo and Abraham Matta meticulously screened my initial thesis abstract and draft, and provided insightful feedback.

Cyrus Shaoul, Henry Minsky, and Milan Minsky provided me with the opportunity to work on cutting-edge Artificial Intelligence technologies.

Patrick Henry Winston introduced me to Artificial Intelligence and treated me as a fellow traveler when I knew nothing about AI.

All my colleagues, collaborators, and friends with whom I have had countless brainstorming sessions on everything, I will always remember them, their wisdom, and our conversations.

I am above all grateful for everyone who has directly and indirectly contributed to shaping me into what I am, and for all the opportunities that I have been fortunate enough to encounter.

DISAMBIGUATING NATURAL LANGUAGE VIA ALIGNING MEANINGFUL DESCRIPTIONS

YIDA XIN

Boston University, Graduate School of Arts & Sciences, 2022

Major Professor: Sang Peter Chin, PhD
Research Professor of Computer Science

ABSTRACT

Artificial Intelligence (AI) technologies are increasingly pervading aspects of our lives. Because people use natural language to communicate with each other, computers should also use natural language to communicate with us. One of the principal obstacles to achieving this is the ambiguity of natural language, evidenced in problems such as prepositional phrase attachment and pronoun coreference. Current methods rely on the statistical frequency of word patterns, but this is often brittle and opaque to people.

In this thesis, I explore the idea of using commonsense knowledge to resolve linguistic ambiguities. I introduce PatchComm, which invokes explicit commonsense assertions to solve context-independent ambiguities. When commonsense assertions are missing, I invoke RetroGAN-DRD, which leverages state-of-the-art inference techniques such as retrofitting and generative adversarial networks (GAN) to infer commonsense assertions. I build upon that with ProGeneXP, which brings state-of-the-art language models to the task of describing its inputs and implicit knowledge in natural language while providing meaningful descriptions for PatchComm to align to further resolve linguistic ambiguities. Finally, I introduce DialComm to lay the groundwork

for moving from single-sentence disambiguation to discourse. Specifically, DialComm builds upon PatchComm to obtain information from single sentences and integrates such information with additional commonsense assertions to build integral frame representations for discourses. I illustrate DialComm's ability with an application to end-user programming in natural language.

The contributions of this dissertation lie in showing how commonsense inference can be integrated with parsing to resolve ambiguities in natural language, in a transparent manner. I have implemented three candidate systems, with increasingly sophisticated approaches. I verified that they perform well on some standard tests, and they operate in such a way that is understandable to people. This obviates the mythical inevitability of an interpretability-performance tradeoff. I have shown how my techniques can be used in a candidate application, programming in natural language.

My work leaves us in a good position to exploit further advances in natural language understanding and commonsense inference. I am optimistic that natural, transparent communication with computers will help make the world a better place.

Contents

List of Tables

List of Figures

List of Abbreviations

AGI	.....	Artificial General Intelligence
AI	.....	Artificial Intelligence
ASI	.....	Artificial Super Intelligence
BERT	.....	Bidirectional Encoder Representations from Transformers
BLEU	.....	Bilingual Evaluation Understudy
BoW	.....	Bag-of-Words
C660	.....	Cambridge Rare Words
CBoW	.....	Continuous Bag-of-Words
CPU	.....	Central Processing Unit
CSKB	.....	Commonsense (Common Sense) Knowledge Base
DL	.....	Deep Learning
DRD	.....	Deep Relationship Discovery
GAN	.....	Generative Adversarial Network
GPT	.....	Generative Pre-trained Transformer
GRU	.....	Gated Recurrent Unit
HCAI	.....	Human-Centered Artificial Intelligence
LM	.....	Language Model
LM-GAN	.....	Language Model Generative Adversarial Network
LSTM	.....	Long Short-Term Memory
ML	.....	Machine Learning
NL	.....	Natural Language
NLP	.....	Natural Language Processing
NLU	.....	Natural Language Understanding
NMT	.....	Neural Machine Translation
OOP	.....	Object-Oriented Programming
PL	.....	Programming Language
PP	.....	Prepositional Phrase
RNN	.....	Recurrent Neural Network
Seq2seq	.....	Sequence-to-Sequence
SL	.....	SimLex
SOTA	.....	State-of-the-Art and State of the Art
SV	.....	SimVerb
UI	.....	User Interface
WSC	.....	Winograd Schema Challenge
WWW	.....	World Wide Web
XAI	.....	Explainable Artificial Intelligence

Chapter 1

Introduction

1.1 Toward Transparent Natural Language Processing in Artificial Intelligence

Artificial Intelligence (AI) in recent years has been hailed by many as a major wave of technological revolution, thanks to the field's awakening to and concentration around Deep Learning (DL) since over a decade ago. Since then, there have been major efforts in academic research on and industrial applications of Deep Learning models in vital areas such as Computer/Robot Vision (CV/RV) and Natural Language Processing (NLP), where computers/robots are programmed to use Deep Learning models to learn to perceive our world and understand our languages.

The continual successes of Deep Learning models on one benchmark after another, have hypnotized and intoxicated many Deep Learning enthusiasts but irritated many Deep Learning critics, both inside and outside of the AI community. Most disagreements between enthusiasts and critics revolve around the notion of AI *performance*, along these two main dimensions:

1. Will current AI accelerate towards Artificial General Intelligence (AGI)?

2. Is AI good or bad? Will AGI be good or bad?

Along the first dimension, admittedly AGI is not a well-defined term. Still, most AI experts and dilettantes agree that "AGI" at the very least means AI systems that possess human-like or human-level intelligence. For example, we expect AGIs to able

to learn a diverse range of knowledge from, and solve a wide variety of problems in, the world as we know it. Current state-of-the-art AI models are generally good at neither domain-general knowledge acquisition nor domain-general problem solving. These models do indeed acquire sophisticated knowledge at different levels of specificity, from the benchmark tasks on which they have been trained. However, the knowledge they learn have been shown to be domain-narrow and brittle. Domain-narrow, because models trained on one category of tasks generally cannot be directly applied to a different category, until it has been significantly fine-tuned or even re-trained for the new category. Brittle, because even within the same category, models can easily be fooled when they are set to apply their acquired knowledge to new tasks. Based on these observations, I believe that current AI systems lack essential ingredients to accelerate toward AGI. One such ingredient, I believe, is *commonsense*. Consequently, in this dissertation, I devote a large amount of attention to commonsense. Specifically, I discuss the role of commonsense knowledge and inference in natural language processing, NLP.

Along the second dimension, there are of course many variables at play. Given the state of AI, in order to answer whether AI is good or bad, we should always first ask who is using AI and how they are using it. Clearly, if bad people use AI, the outcome is predictably bad. The main challenge of AI deployment, however, is that even if good people use AI — as I do believe is vast majority of the times — the outcome might still turn out bad. In this dissertation, I delegate the topic of *bad people doing AI* to humanitarian experts. Instead, I point to the issue of well-intentioned people unintentionally doing "bad" AI. To that end, I once again call upon the use of commonsense to mitigate this issue, in addition to the "merely academic" concern of applying commonsense to language understanding.

Throughout my discussions on commonsense, I will reiterate the notion of AI

transparency and *interpretability*. I argue that the best way forward is to think about performance and interpretability in a symbiotic relationship, rather than a trade-off relationship. Fortunately, commonsense is the key substrate that underlies both performance and interpretability.

My idea for implementing the said symbiotic relationship between AI performance and interpretability, is implementing systems that use commonsense for problem solving and making sure that their internal procedures and data representations are transparent to us. This way, they can explain themselves to us. The best interface for this kind of transparency and interpretability is of course our human natural language, and this is why my dissertation focuses on discovering transparent solutions to ongoing problems in NLP. Consequently, in this dissertation, I implement my idea.

1.2 Organization

The organization of this document is as follows:

In chapter 2, I discuss PatchComm (Xin et al., 2021), a general-purpose system that leverages large-scale commonsense knowledge in both symbolic and vectorized forms, to help syntactic parsers resolve such prominent syntactic ambiguities as prepositional-phrase attachment ambiguity and pronoun coreference ambiguity. PatchComm is transparent from the ground up, because it uses commonsense knowledge and inference mechanisms in an explicit way and reports which exact pieces of knowledge have been used. In addition, we will see how PatchComm achieves a performance boost while ensuring its basic transparency, by using more sophisticated neural network inference mechanisms.

In chapter 3, I discuss ProGeneXP, an instructable system that aims at bringing out transparency from large-scale language models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) who are notoriously opaque. ProGeneXP is open

to receiving new instructions from humans at all times, and is capable of generating natural language descriptions on the fly to showcase its learning and its understanding of the world to human users in natural language. To evaluate the quality of such descriptions in automated tasks, I will revisit the task of pronoun coreference disambiguation. We will see how ProGeneXP uses its self-generated natural language descriptions to provide transparency for human users, and to improve its performance on downstream tasks upon baselines.

In chapter 4, I discuss DialComm, a general-purpose system that builds on PatchComm's ability to improve sentence parsing and enables language understanding at the discourse level. The key is to leverage both discourse context and readily available commonsense to build frame (Minsky, 1974) representations of discourses. DialComm, like PatchComm, is transparent from the ground up for the same reason. We will see, via demonstrations, that DialComm can construct frames from discourses, use the frames to go back to sentence-level ambiguities and resolve them, and also use the frames to lay the groundwork for the downstream task of end-user programming in natural language.

In chapter 5, I review general related work on natural language processing and commonsense inference, setting the stage for a discussion on future work.

Finally, in chapter 6, I summarize my contributions and postulate future directions for continuing to make progress on the issue of natural language disambiguation. I land that chapter and this dissertation stating optimistically that the future of AI, where humans and machines have healthy communications with each other, is one to look forward to.

Chapter 2

PatchComm

2.1 PatchComm: Using Commonsense Knowledge to Guide Syntactic Parsers

PatchComm (Xin et al., 2021) is a general-purpose, modular system that leverages commonsense to resolve context-independent syntactic ambiguities. In developing PatchComm, we have targeted specifically at *prepositional phrase* (PP) *attachment* ambiguities and *pronoun coreference* ambiguities, and investigated how the use of commonsense is a natural solution to resolving these ambiguities.

In its latest version since the work of (Xin et al., 2021), PatchComm consists of a syntactic parsing module and a commonsense module, shown in Figure 2·1.

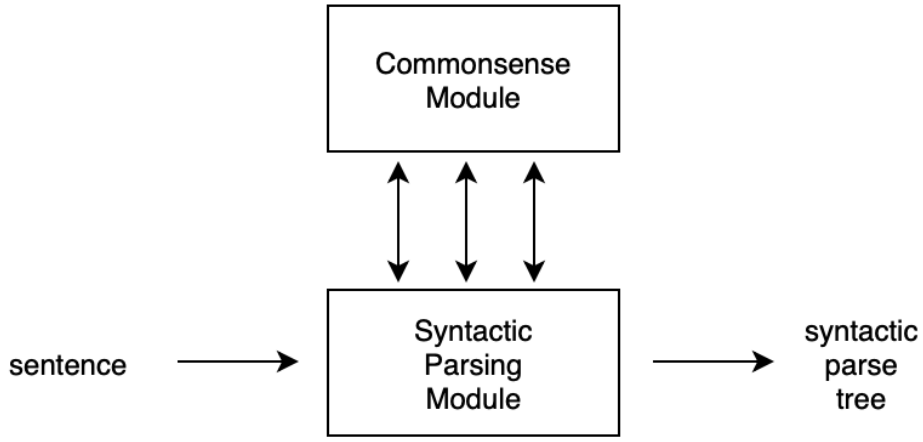


Figure 2·1: Architecture of PatchComm.

Both modules can be instantiated, respectively, by any syntactic parser or com-

monsense reasoning system of choice. In its current implementation, PatchComm uses spaCy dependency parser[1] (Honnibal et al., 2020), including its plug-in module, NeuralCoref[2] (Clark and Manning, 2016b; Clark and Manning, 2016a), for the syntactic parsing module; and uses the large-scale ConceptNet knowledge base [3] (Speer et al., 2017) for the commonsense reasoning module.

2.1.1 Syntactic Parsing

Syntactic parsing is the linguistic process of analyzing the grammatical and syntactic information of a sentence and compiling such information into a *syntactic parse tree*, as allude to in Section 1.2 of Chapter 1.

The two most prevalent paradigms of syntactic parsing are *constituency* and *dependency*. In both kinds of parsing, the tree-constructing process begins with the words in a given sentence. The difference is that, in constituency parsing, the process begins by grouping together related words into higher-level phrases, commonly represented as nodes in a tree; this process then repeats recursively to build higher- and higher-level nodes, until the whole sentence is represented as one top-level node. The end result is a complete tree, called *constituency parse tree*, made up of all the words and intermediate nodes obtained from the processing. On the other hand, in dependency parsing, the process begins by selecting a specific word called the *root*, which is treated as the top-level node for the resulting parse tree. Then, the root node is treated as *head*, and child-head relations are invoked to find all the *child nodes* (a.k.a. *children*) of the root node. After that, each child is then treated as a head, and the process repeats recursively, until all paths terminate in words of the sentence. The end result is a fully-connected tree whose nodes are all the words of the sentence and edges are directed from head to child and labeled with *syntactic depen-*

[1]Currently version 2.3.5.
[2]Currently version 4.1.0. See https://github.com/huggingface/neuralcoref
[3]Currently version 5.8. See http://blog.conceptnet.io/

dency labels. Roughly speaking, one may think of constituency parsing as bottom-up and dependency parsing as top-down, but this is only accurate in a sense.

Figure 2.x shows an example of constituency and dependency parse trees for a given sentence, produced by CoreNLP (Manning et al., 2014) and spaCy (Honnibal et al., 2020), respectively.

In its current version, PatchComm makes use of the spaCy dependency parser, along with spaCy's built-in NeuralCoref (Clark and Manning, 2016b; Clark and Manning, 2016a). In section 2.3, we discuss details on PatchComm's basic set of operating rules for identifying prepositional phrases and pronouns that are to be disambiguated, as well as their surrounding syntactic constraints in given sentences. That said, we note that PatchComm is a general-purpose system that can be implemented with any syntactic parser of any style of choice, so long as its basic set of operating rules are written in the selected parser's native vernacular. Of course, one might argue that having to re-implement such a set of rules defeats the claim that PatchComm is general-purpose. We disagree, because even though implementation details change, the underlying syntactic constraints and modular design remain the same. Still, we recognize that it is a good idea to try to automate the process of mapping from implementations in spaCy's native vernacular to a new parser's native vernacular — perhaps using Machine Learning, possibly with the help of Codex (Chen et al., 2021) (see Chapter 4) — so long as the automation process itself is interpretable, too.

2.1.2 Sentence-Level Syntactic Ambiguities

Syntactic parsers primarily work on individual sentences and are frequently hindered by syntactic ambiguities from reliably producing syntactic parse trees that reflect the intended meaning of their input sentences. Because PatchComm builds around syntactic parsers, it makes sense that PatchComm specifically targets at addressing sentence-level ambiguities. Among the most notorious sentence-level ambiguities are

prepositional phrase attachment ambiguities and *pronoun coreference* ambiguities.

Roughly speaking, whenever we look at standalone sentences without contexts, we may separate them into two categories:

1. Context-Independent Sentences:

   These are unambiguous for humans but assumed by default ambiguous for machines. To resolve ambiguities in such sentences, we humans can draw on our commonsense knowledge and zero in on the correct disambiguation solution, all within the split of a second.

2. Context-Dependent Sentences:

   These are ambiguous for both humans and machines; such sentences must be embedded in larger contexts to be correctly understood, even by humans and certainly by machines.

PatchComm only targets at resolving context-independent ambiguities, and I spend the remainder of this section (section 2) describing *how* PatchComm has been developed for that target; I delegate the discussion on context-dependent ambiguities to the next chapter. Overall, PatchComm is a general-purpose framework that sets the groundwork for leveraging explicit commonsense knowledge to resolve context-independent ambiguities. PatchComm'a nature of leveraging commonsense knowledge in explicit manners, is why I dub it as a *transparent* approach.

2.1.3 Commonsense Knowledge and Reasoning

To be sure, there are prior work that have utilized knowledge to resolve linguistic ambiguities. However, those are *linguistic knowledge* instead of commonsense knowledge. In order for a machine to truly understand natural languages, clearly linguistic knowledge alone are not enough — clearly commonsense knowledge is necessary.

To make it clear that commonsense is a must for language understanding, I have decided to make PatchComm revisit the aforementioned notorious ambiguities: prepositional phrase attachment and pronoun coreference. Through the lens of these concrete problems, I hope to convince the reader of the significance of a commonsense-based solution to not just these problems, but NLP at large.

For PP attachment disambiguation, the works of (Belinkov et al., 2014) and (Nakashole and Mitchell, 2015) are examples where crowdsourced linguistic knowledge in the form of WordNet (Miller, 1995; Fellbaum, 1998), VerbNet (Schuler and Palmer, 2005), and FrameNet (Baker et al., 1998) are used to tackle PP attachment ambiguities. In these work, linguistic knowledge from WordNet, VerbNet, and FrameNet are converted into knowledge vectors; sentences from task data are first decomposed into words and phrases according to predetermined syntactic constraints, then also converted into text vectors. Afterwards, knowledge vectors and text vectors are merged to form *feature vectors*. After that, feature vectors of different words and phrases are further merged, according to reconstruct the aforementioned syntactic constraints (e.g. the constraint provided in (Ratnaparkhi et al., 1994)), into more phrase vectors and sentence vectors. And finally, these word, phrase, and sentence vectors are utilized to determine the "best" attachment candidate for the PP attachment, where "best" means *the most similar* or *the highest probability*, etc., in a classification setting.

For pronoun coreference, there has largely been two different directions:

- The open-ended direction that targets at resolving coreferences in discourses of any size or style;
- The direction focused on Winograd Schema styled sentences, whose solution was initially believed to require a significant amount of human-like or human-level commonsense knowledge about the world.

In the open-ended direction, recently there has been a successful series of work on end-to-end, *span-based* coreference resolution that began with the work of (Lee et al., 2017). For the convenience of the reader, that work made clear the definitions of two keywords:

- *Span*: A span is defined to be a specific sub-sequence of consecutive words taken from a given data point (i.e. discourse). For example, in the sentence "I ate an apple.," *ate an* is a span.

- *Mention*: A mention is a correctly resolved coreferential entity. For example, in the sentence "I ate an apple and it was delicious.," *apple* is the mention of the referring expression *it*.

The coreference task is defined as looking for all the spans in a discourse that correctly correspond to all mentions. In this task setting, all words are once again converted to word vectors, just like in the aforementioned related works on prepositional phrase attachment. To keep things on point, large-scale pre-trained word embeddings such as Word2Vec (Mikolov et al., 2013) and GloVe (Pennington et al., 2014)) have been used. In pre-training such word embeddings, linguistic knowledge sources, such as the aforementioned WordNet, VerbNet, FrameNet, can also be used. Whether pre-trained on explicit knowledge sources or not, word embeddings are believed to encode knowledge from their training sources.

In the Winograd Schema direction, the series of work, starting with the Winograd Schema Challenge (Levesque et al., 2012) and culminating in the WinoGrande dataset for which human-level performance has been observed (Sakaguchi et al., 2020), speaks volumes. Initially, (Levesque et al., 2012) proposed the Winograd Schema Challenge (WSC) as an alternative to Turing Test, because the authors believed that any solution to WSC would surely require extensive commonsense capability of machines. But as it had turned out, one solution seems to "only" require pre-trained large

language models, and it is unclear how much commonsense these models really possess. The series of works culminated in (Sakaguchi et al., 2020), which proposed the large-scale crowdsourced dataset, WinoGrande, accompanied by a dataset debiasing algorithm, called AfLite. The authors then showed that transfer learning (Yosinski et al., 2014) from WinoGrande to other WSC-related datasets boosts performances on those datasets, including a human-level performance of 93.1%. However, in a footnote, the authors admitted that models trained on the AfLite-debiased version of WinoGrande "showed only chance level performance." Another notable point about (Sakaguchi et al., 2020) is that the word embeddings utilized are directly from the most state-of-the-art language models such as BERT (Devlin et al., 2019) and (Liu et al., 2019), which many believe are capable of learning deep and complex semantics from very large-scale corpora.

Indeed, large language models have been shown to be very effective on tasks that had been inconceivably challenge, until these models were developed. However, it is open to debate how much *commonsense* — especially the kind that we humans can make sense of — these models really possess. And of course, on the note of transparency, the consensus of the AI community is that these models are opaque black boxes. When they make mistakes or fail, oftentimes in quite serious ways, people don't know how or why — frequently not even their very developers. Consequently, whenever an enthusiast claims that these large-scale black-box models will of course be friendly to humans and thus we only need to worry about improving their performance and domain-generality, I see no real evidence to support that claim.

In light of making machine-learned knowledge transparent to humans and making machines do commonsense reasoning in human-like ways, I develop PatchComm to contrast with purely vector and neural-network approaches. The contrast is in two main ways:

1. At the highest decision-making level, PatchComm operates in the symbolic space, rather than vector space, of language. As I will discuss in section 2.3, PatchComm is built with a native set of transparent, symbolic rules defined over spaCy's native part-of-speech (PoS) tags, syntactic dependency labels, child-head relations etc.

2. At the lower knowledge-acquisition and reasoning levels, PatchComm is built to utilize state-of-the-art neural network mechanisms for embedding knowledge. However, PatchComm uses such knowledge in a modular, rather than end-to-end, fashion. Explicit scoring functions have been implemented, so that the underlying statistical mechanisms for evaluating knowledge strengths are made explicit and simple.

2.2 Prepositional Phrase Attachment with ConceptNet

In this section, I discuss details on how PatchComm first identifies prepositional phrases from sentences according to predetermined syntactic constraints, and then leverages context-independent commonsense knowledge to disambiguate the prepositional phrases to the best of its ability. In this section, I only discuss the more primitive way of directly querying ConceptNet (Speer et al., 2017) for commonsense knowledge. In section 2.4, I discuss RetroGAN-DRD, a Deep Neural Network (DNN) model that first embeds knowledge and then becomes useful as a commonsense inference module. After that, in section 2.5, I discuss the more advanced method of using RetroGAN-DRD for commonsense inference to ultimately help PatchComm resolve prepositional phrase ambiguities.

2.2.1 Task Setup

By default, PatchComm assumes all encountered prepositional phrases are ambiguous. When there is a single prepositional phrase, PatchComm extracts it according to the basic syntactic constraint of

$$(subj, verb, obj, prep, pobj)$$

from (Ratnaparkhi et al., 1994), where *subj* means subject, *verb* means verb, *obj* means object, *prep* means preposition, and *pobj* means prepositional object. All of $(subj, verb, obj)$ are candidates to which the prepositional phrase is to be attached; $(prep, pobj)$ is the minimally-simplified prepositional phrase that has been rid of any determiner or adjective. In general, for multiple prepositional phrases — e.g. N of them — PatchComm simply extends the basic syntactic constraint into

$$(subj, verb, obj, prep_1, pobj_1, \cdots, prep_N, pobj_N)$$

2.2.2 Implementation Details

PatchComm first obtains spaCy's outputs, and then uses commonsense knowledge from ConceptNet (Speer et al., 2017) to check spaCy's attachment decision. If ConceptNet deems a spaCy decision wrong, then PatchComm modifies that decision accordingly. Namely, PatchComm first obtains spaCy's attachment decision, which is attaching the prepositional phrase to either *verb* or *obj*. The attachment to *verb* indicate, in fact, a semantic association with with *subj* or *verb*, or any combination of both. As a result, PatchComm proceeds to query ConceptNet with the pairs (*subj*, *pobj*) and (*verb*, *pobj*) to determine the strength of evidence for attaching to *verb*; and the pair (*obj*, *pobj*) to determine the strength of evidence for attaching to *obj*.

As of version 5 and beyond, ConceptNet is shipped with assertions weights that are learned from a previous project called *AnalogySpace* (Speer et al., 2008). Consequently, to seek strengths of evidence when querying ConceptNet with the aforementioned pairs, PatchComm simply relies on the acquired assertion weights by AnalogySpace. Specifically, PatchComm first looks at a given pair, finds all combinations of all lexical variants of both concepts in that pair, and returns the max-weight assertion as the result of querying ConceptNet for that pair. PatchComm does this for all pairs. Then, PatchComm simultaneously

- Looks at the pairs $(subj, pobj)$ and $(verb, pobj)$, and chooses the higher-weight assertion between the two;
- Looks at the pair $(obj, pobj)$.

If the assertion weight for $(subj, pobj)$ and $(verb, pobj)$ is higher than the assertion weight for $(obj, pobj)$, PatchComm decides that the prepositional phrase attaches to *verb*; otherwise, PatchComm decides to *obj*. Because the assertion weights are represented as floating-point numbers, in practice, we almost see that these two weights

are exactly equal.

Figure 2·2 shows an example of prepositional phrase attachments where the sentence has only one prepositional phrase. spaCy disambiguates without (above) and with (below) the help of PatchComm.

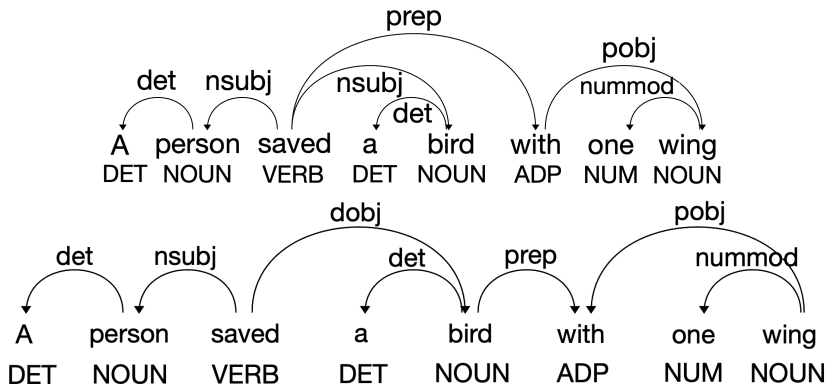


Figure 2·2: spaCy's (above) versus PatchComm's PP attachment decisions, where it is obvious that PatchComm's decision makes more *common*sense.

In the aforementioned general case where a sentence has N prepositional phrases, PatchComm solves all prepositional phrases from left to right, one at a time, in a fashion similar to that for the single case above. Specifically, for $n \in [1,\cdots,N]$, every time after PatchComm resolves the n-th prepositional and moves on to the $(n+1)$-th prepositional phrase, PatchComm now treats all of $(subj, verb, obj, pobj_1,\cdots,pobj_n)$ as candidates for the attachment of $(prep_{n+1}, pobj_{n+1})$. PatchComm repeats this process until all N prepositional phrases have been attached.

Figure 2·3 shows an example of prepositional phrase attachments where the sentence has more than one prepositional phrases — in the specific example, there are two prepositional phrases. spaCy disambiguates without (above) and with (below) the help of PatchComm.

It is worth noting that, when queried with a pair of concepts (c_1, c_2), ConceptNet returns both all assertions that start with c_1 and end with c_2 and all assertions

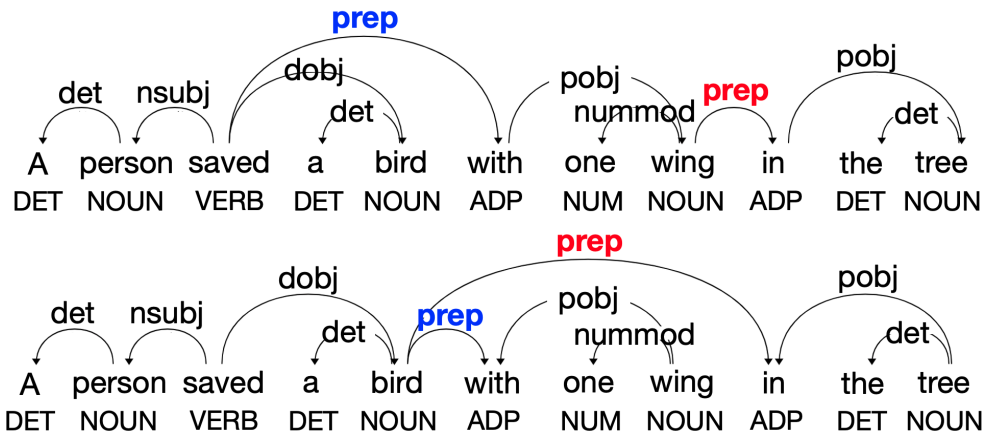


Figure 2·3: spaCy's (above) versus PatchComm's PP attachment decisions, where it is obvious that PatchComm's decision makes more *common*sense.

that start with c_2 and end with c_1. Also, because ConceptNet is not omniscient, it frequently happens that ConceptNet does not have any assertion for a given pair of concepts and PatchComm cannot get anything. Whenever this happens, PatchComm defaults back to spaCy's attachment decisions.

2.2.3 Experiments and Results

For experiments on prepositional phrase attachment, I mainly conducted two experiments.

For my experiment, I created a set of 100 sentences that contains only two attachment candidates. The baseline for this experiment is spaCy itself. Below, table 2.1 shows the result from the experiment:

	spaCy	spaCy + PatchComm w/ ConceptNet
Self-created	57.0%	67.0%

Table 2.1: Result of PatchComm with ConceptNet on self-created prepositional phrase attachment dataset, compared with spaCy alone.

2.3 Pronoun Coreference

In this section, I discuss details on how PatchComm first identifies pronouns from sentences according to predetermined syntactic constraints, and then leverages context-independent commonsense knowledge to disambiguate the prepositional phrases to the best of its ability. In this section, I only discuss the more primitive way of directly querying ConceptNet (Speer et al., 2017) for commonsense knowledge. In section 2.6, I discuss the more advanced method of using RetroGAN-DRD for commonsense inference to ultimately help PatchComm resolve pronoun coreference ambiguities.

2.3.1 Task Setup

Just like how the prepositional phrase attachment task is classified into single-PP and multi-PP, here I also classify the pronoun coreference task into single-pronoun and multi-pronoun. Also, just like before where I assume that each prepositional phrase can only refer to an entity that precedes it, here I also assume that each pronoun can only refer to an entity that precedes it. In general this is not true, but for simplicity, this assumption will suffice.

PatchComm translates the criteria specified in (Levesque et al., 2012) into the basic syntactic constraint of

$$(\cdots\ ent_1\ \cdots\ ent_2\ \cdots\ pron\ \cdots)$$

for the single-pronoun case, where ent_i is a candidate entity and $pron$ is the pronoun to be resolved. And for the multi-pronoun case, note that the sequence of entities can interweave with the sequence pronouns, so to pinpoint to a specific pronoun, the constraint looks like

$$(\cdots\ ent_1\ \cdots\ ent_M\ \cdots\ pron_n\ \cdots)$$

where PatchComm has to determine the best of the M entities that the n-th pronoun should corefer to.

2.3.2 Implementation Details

The key to PatchComm's pronoun coreference mechanism is finding the right *description token* for the ambiguous pronoun. This turns out to be a very difficult step. In this chapter, I discuss only the most primitive mechanism for finding description token that has been around since the very first version of PatchComm. But in the next chapter, I discuss a sophiscated mechanism that takes advantage of state-of-the-art Deep Neural Network language models. This more sophiscated mechanism is inspired by not only the specific problem of pronoun coreference, but also AI transparency and self-explainability in general.

In the current version, a pronoun *description token* is defined as the token in the sentence that recursively shares the same head with the pronoun, marking it as a *description* token. If the immediate child is neither an entity nor a property, then we keep searching its children, until we get to either an entity or a property.

After finding the descriptor token, PatchComm queries ConceptNet with the pairs ($Entity_1$, *description*) and ($Entity_2$, *description*). When a sentence has multiple ambiguous pronouns, PatchComm assumes that each pronoun can refer to any entity that occurs in the sentence, and uses the same description-token-matching mechanism detailed above.

Figure 2·4 shows examples of pronoun coreference performed by NeuralCoref without PatchComm. Figure **??** shows how PatchComm helps improve some outputs in Figure **??**. Note in Figure 2·5 that PatchComm embeds coreference information into spaCy-style dependency parse, as shown by the red highlighted *coref* label.

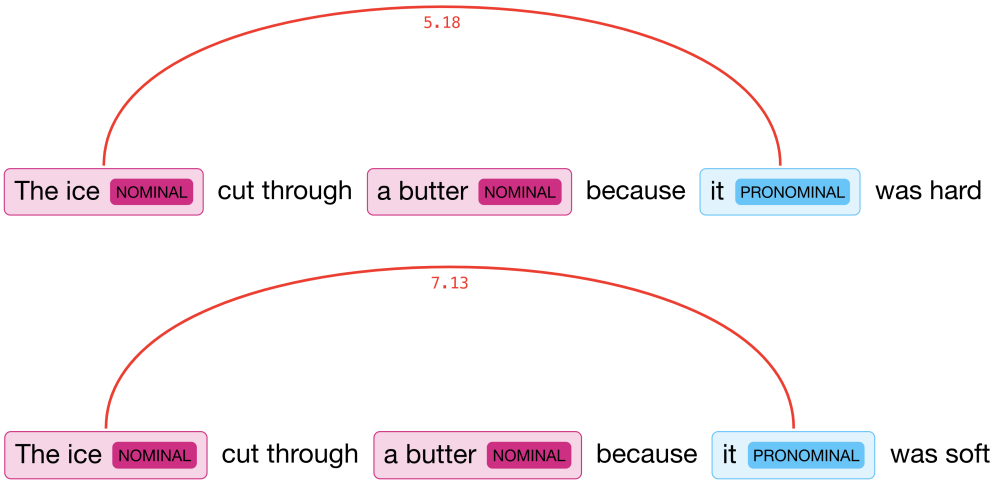


Figure 2·4: NeuralCoref when faced with ambiguous pronouns that should be resolved differently.

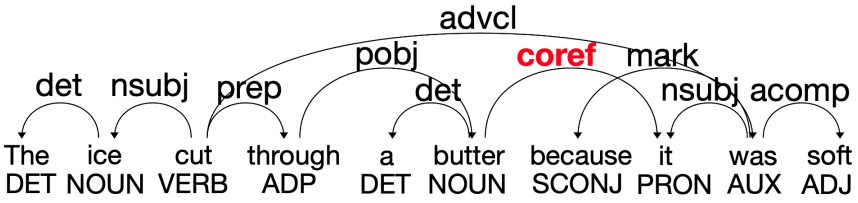


Figure 2·5: PatchComm resolves the ambiguity from Figure 2·4.

2.3.3 Experiments and Results

For my experiment, I used the publicly available WSC273 dataset[4] from (Levesque et al., 2012). Below, table 2.2 shows the result from these two experiments. I chose NeuralCoref (Clark and Manning, 2016b; Clark and Manning, 2016a) to be the baseline for both experiments, to abide by the setting of syntactic parsing.

[4]Full dataset of (Levesque et al., 2012) is available here: `https://cs.nyu.edu/~davise/papers/WinogradSchemas/WS.html`. Note that this dataset actually consists of 285 sentences, but the last 12 sentences are thoughts to be "less good" than the original 273. I only used the first 273 sentences.

	NeuralCoref	NeuralCoref + PatchComm w/ ConceptNet
WSC273	31.1%	39.9%

Table 2.2: Result of PatchComm with ConceptNet on WSC273 dataset, compared with NeuralCoref alone.

2.4 RetroGAN-DRD

To recap, in sections 2.2 and 2.3, I discussed my attempt to directly apply readily available commonsense knowledge, basically in their native symbolic form, to the aforementioned syntactic ambiguities. However, there are two immediately obvious shortcomings:

- The task setups may not be the optimal models for the actual underlying problems.
- ConceptNet cannot possibly cover all commonsense knowledge ever; and even supposing it could, in their native symbolic form, matching the right kinds of knowledge for a task-at-hand is a serious problem that requires too much commonsense in the first place.

From all the assumptions that I made in the last two sections, the reader can see that the task modeling can definitely be scaled up and improved. Much of this is, admittedly, future work that I currently do not know how to shed more light upon. In this section, I introduce RetroGAN-DRD, which targets at the second of the aforementioned two problems.

RetroGAN-DRD starts with the work of (Colon-Hernandez et al., 2019), which first-and-foremost identifies that neural commonsense inference consists of two steps:

- Integrated Semantics:

  Somehow, we need to find a way to add knowledge "flavors" into plain words or concepts. One way to do this is to first turn both knowledge and plain words into vectors, thus unifying their representation; then apply mathematical techniques on both kinds of vectors, so that the best of both worlds are integrated.

- Commonsense Inference:

  Somehow, we need to find a way to draw assertions that connect a given pair of concepts in some vector space. One way to do this is to train Machine Learning models that can predict the likelihoods and compute the strengths of all possible assertions that connect the concepts of interest, and then determine which assertions are the "best."

The clear separation of these two stages suggest a modular approach. Consequently, RetroGAN has been introduced for the first stage, and Deep Relationship Discovery (DRD) for the second stage. In a nutshell, RetroGAN looks at pairs of plain word embeddings and their knowledge-flavored counterparts, and learns a mapping from the plain embeddings to the knowledge-flavored embeddings; the same mapping can then be applied to plain embeddings that RetroGAN has not yet seen, to generate knowledge-flavored embeddings. This allows RetroGAN to go beyond a finite knowledge source and reason about out-of-knowledge concepts. DRD subsequently takes a pair of knowledge-flavored concepts (which are the outputs of RetroGAN), extracts higher-level semantics out of this pair, and uses the higher-level semantics to determine the strength for each possible assertion.

2.4.1 RetroGAN

RetroGAN consists of two main stages:

1. Specialization via *Retro*fitting:

   In retrofitting, off-the-shelf plain word embeddings are adjusted according to semantic constraints from knowledge graphs. In other words, the inputs are word embeddings and knowledge embeddings, and the inputs are knowledge-adjusted (or knowledge-flavored) word embeddings. It is commonly accepted that word embeddings capture distributional semantics and knowledge embeddings capture knowledge-graph semantics. Consequently, it is believed that knowledge-flavored embeddings capture of the best of both worlds.

2. Post-Specialization via *GAN*:

   GAN stands for Generative Adversarial Network. GAN has been a well-studied topic of Machine Learning and GAN architectures have been well-developed to deployed to a wide variety of real-world applications. For the purposes of RetroGAN, GANs help with learning the aforementioned mapping from plain word embeddings to retrofitted word embeddings, which is in turn directly useful for out-of-knowledge inference.

Retrofitting is first proposed in (Faruqui et al., 2015) as a technique where word embeddings trained on large-scale corpora are adjusted according to semantic constraints taken directly from some (also large-scale) knowledge graphs. The end result is that the plain word embeddings are mapped from their original vector space onto a new vector space of knowledge-flavored word embeddings. The knowledge-flavored word embeddings are better *aligned* in that similar words are more "parallel with" one another. This kind of alignment reflects the impact of the informing knowledge graphs. More recently, the work of (Vulić and Mrkšić, 2018) proposes a new

retrofitting mechanisms called Attract-Repel, which not only aligns similar words, but also adjusts dissimilar words so that they are more "orthogonal to" one another.

RetroGAN takes advantage of ATTRACT-REPEL. In the context of commonsense inference, as inspired by the work of (Ponti et al., 2018), the retrofitting process is re-dubbed as *specialization*.

In practice, retrofitting and specialization have been shown quite effective, e.g. ConceptNet NumberBatch (Speer et al., 2017). However, retrofitting suffers from one significant shortcoming: it only works on concepts that already exist within semantic constraints, be it in the form of knowledge graphs as in ConceptNet, lists of synonyms and antonyms as in Attract-Repel, or the like. To go into the more open-ended world of linguistic concepts, a commonsense inference module must be able to reason about concepts outside of the the knowledge source. Consequently, the same work of (Ponti et al., 2018) also introduces *post-specialization*, which takes place after the specialization/retrofitting stage and targets precisely at the out-of-knowledge problem. During post-specialization, a GAN is trained to learn a mapping from plain embeddings to retrofitted embeddings for in-knowledge words, and then applied on plain embeddings of out-of-knowledge words to generate "knowledge-flavored" embeddings for those words.

Architecture of RetroGAN

Whereas (Ponti et al., 2018) uses a one-directional GAN much like the basic one proposed in (Goodfellow et al., 2014), RetroGAN utilizes a CycleGAN-like (Zhu et al., 2017) architecture, in order to take advantage of CycleGAN's *cycle consistency* property. During training, both the forward and backward mappings are learned, but during generation of out-of-knowledge embeddings, evidently, only the forward mapping is applied. Figure 2·6 shows the complete architecture of RetroGAN.

RetroGAN crucially consists of a CycleGAN, which is essentially two GANs that

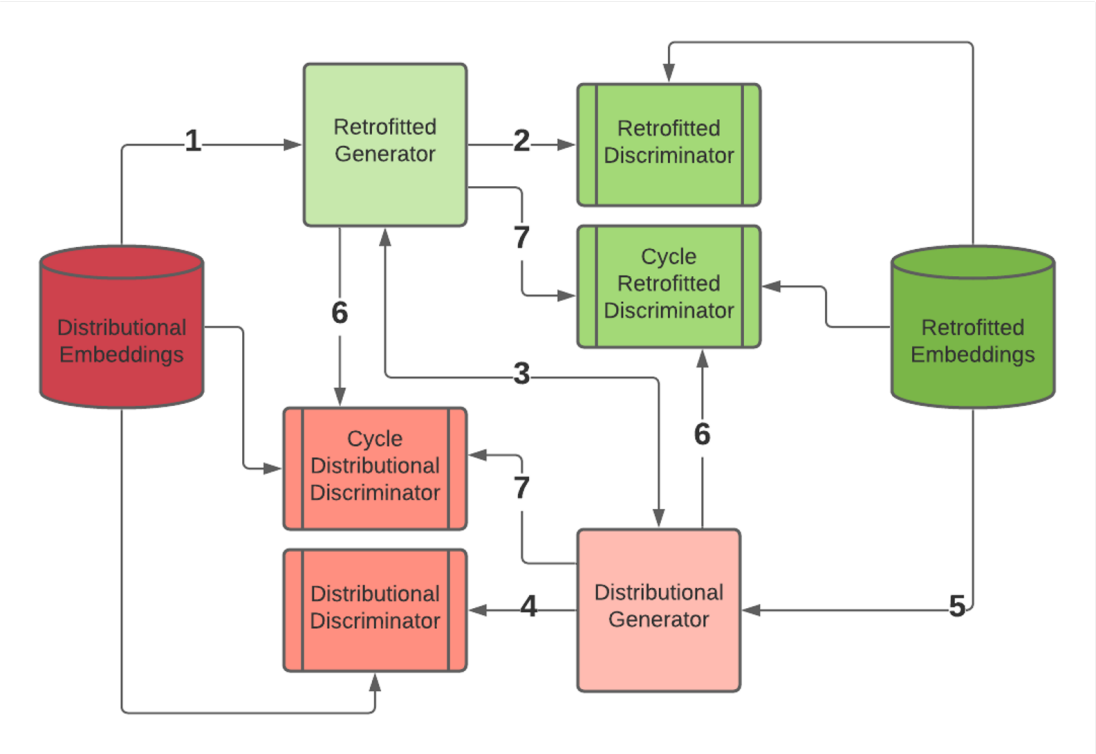


Figure 2·6: Architecture of RetroGAN. Note its cyclic nature.

cooperate to balance a combination of losses to transform a particular word embedding $x_i \in X$ from its original domain X to its counterpart $y_i \in Y$ in the retrofitted domain Y, and vice-versa. In both GANs that we employ, the generator consists of an input layer followed by 2 hidden dense layers with 2048 neurons and each followed by a dropout layer (with a percentage of 0.2 for the dropouts), and a final linear output layer with the same dimensionality as the input. The output of this layer, for the trained $G : X \rightarrow Y$ produces the post-specialized embeddings (i.e. a batch of 32 FastText embeddings produces 32 post-specialized embeddings). The hidden layers employ the ReLU (Nair and Hinton, 2010) activation function. Our discriminators have a similar structure (an input layer, 2 hidden layers with dropout but a percentage of 0.3), however, the second hidden layer is followed by a batch normalization layer and the output is a single neuron with a sigmoid activation. The reason for

the batch normalization layer was to stabilize the training. We also utilized a third and fourth conditional discriminator following (Tripathy et al., 2018), to leverage the cyclic architecture on paired data.

A novelty in RetroGAN is the combination of cyclic and non-cyclic optimization objectives: the regular adversarial loss for both GANs (L_{GAN}); the cyclic loss for both generators (L_{CYC}); the identity loss for both generators (L_{ID}); the max margin loss similar to (Weston et al., 2011; Ponti et al., 2018) for both the generators and additionally for the cycle of generators (L_{MM}); and the conditional cycle consistency loss (L_{cCYC}) introduced in (Tripathy et al., 2018). The combined objective has the following form:

$$\begin{aligned} L(G, F, D_X, D_Y) = & L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, X, Y) \\ & + \lambda L_{CYC}(G, F) \\ & + \gamma L_{ID}(G, F, X, Y) + L_{MM}(G, F, X, Y) \\ & + \varsigma L_{cCYC}(G, F, D_{cX}, D_{cY}, X, Y) \end{aligned}$$

where $G : X \rightarrow Y$ is the generator that maps the source domain X of plain word embeddings to the target domain Y of retrofitted word embeddings; $F : Y \rightarrow X$ is the generator that does the opposite; D_X and D_Y are the discriminators for the corresponding domains; and D_{cX}, D_{cY} are our cycle conditional discriminators. For brevity, we only go into details on L_{MM} and L_{cCYC}. The other losses are the standard ones found in their respective works: L_{GAN} is the adversarial loss from (Goodfellow et al., 2014). L_{CYC} is the cycle consistency loss from (Zhu et al., 2017) with a scaling factor of λ (set to 1.0); and L_{ID} is the identity loss from (Zhu et al., 2017), which we scale with γ (set to 0.01). L_{ID} serves as a check of whether the embedding is already in the correct domain. L_{MM} is the max margin loss with random confounders as used

by (Ponti et al., 2018), and as a novel aspect, we add a cyclic margin loss:

$$\begin{aligned} L_{MM}(G,F,X,Y) = \Sigma_{i=1}^{||x||}\Sigma_{j=1|j\neq i}^{k}\tau[&(\delta_{MM} - cos(G(x_i),y_i) + cos(G(x_i),y_j)) \\ &+ (\delta_{MM} - cos(F(y_i),x_i) + cos(F(y_i),x_j)) \\ &+ \boldsymbol{(\delta_{MM} - cos(G(F(y_i)),y_i) + cos(G(F(y_i)),y_j))} \\ &+ \boldsymbol{(\delta_{MM} - cos(F(G(x_i)),x_i) + cos(F(G(x_i)),x_j))}] \end{aligned}$$

Intuitively, this equation makes generated embeddings similar to their gold-standard and different from confounders. RetroGAN further enforces this constraint across the cycle. And lastly, we have the conditional cycle loss, L_{cCYC} (Tripathy et al., 2018)[5], which we scale with ς (set to 1):

$$\begin{aligned} L_{cCYC}(G,F,D_{cX},D_{cY},X,Y) = &\mathbb{E}_{x\sim p_{data}}[log(D_{cX}(G(x),x))] \\ &+ \mathbb{E}_{x\sim p_{data}}[log(1 - D_{cX}(G(x),F(G(x))))] \\ &+ \mathbb{E}_{y\sim p_{data}}[log(D_{cY}(F(y),y))] \\ &+ \mathbb{E}_{x\sim p_{data}}[log(1 - D_{cY}(F(y),G(F(y))))] \end{aligned}$$

Experiments and Results of RetroGAN

For training, we use the Adam Optimizer (Kingma and Ba, 2015) with a learning rate of $5e-5$ for the generators and $1e-4$ for the non-conditional discriminators. We do not train the discriminators used in the regular GAN loss, and instead train the ones in the conditional cycle consistency loss. We also note that we did not perform explicit fine tuning of the scaling parameters, but we will do so in future work through a grid search. We train for 312,500 mini-batches which is the equivalent to the AuxGAN (Ponti et al., 2018) training, using a batch size of 32.

For tests, we use the English Common Crawl FastText with sub-word information

[5]In future work, we will additionally incorporate the paired conditional adversarial loss.

(FT-CC) (Bojanowski et al., 2017) and Numberbatch 19.08 (NB) (Speer et al., 2017) to see how performance would be affected by using embeddings that were already retrofitted with a large KB. We ran ATTRACT-REPEL[6] (Vulić and Mrkšić, 2018) on all these embeddings, and then perform Post-Specialization tests on learning the mapping from FT-CC to the resulting retrofitted embeddings.

As shown in Table 2.3, we run the word similarity benchmarks: SimLex (SL) (Hill et al., 2015) SimVerb (SV) (Gerz et al., 2016), and the Cambridge Rare Word (C660) dataset (Pilehvar et al., 2018). In the Disjoint setting, we evaluate words that are **not seen** in the constraints; in Full, we evaluate words that **are seen** in constraints.

	Disjoint				Full					
	FT-CC, A-R		FT-CC, A-R+NB		FT-CC, Attract-Repel			FT-CC, A-R+NB		
Models	SL	SV	SL	SV	SL	SV	C660	SL	SV	C660
Distributional	0.4644	0.3649	0.4499	0.3643	0.4644	0.3649	0.2973	0.4499	0.3643	0.1068
Attract-Repel	0.4644	0.3649	0.4499	0.3643	**0.7790**	**0.7632**	0.3768	0.7748	**0.7667**	0.2203
AuxGAN	**0.6127**	0.4641	0.6116	0.5331	0.6901	0.5756	0.3899	0.6565	0.5872	0.2088
RetroGAN	0.6028	**0.4702**	**0.6648**	**0.5971**	0.7717	0.7192	**0.5240**	**0.7960**	0.7483	**0.5581**

Table 2.3: Results for SimLex and SimVerb; best values in **bold font**.

We trained AuxGAN for 10 epochs of 1M iterations (which in AuxGAN is a single embedding pair rather than a batch of pairs) with both plain stochastic gradient descent (SGD) and Adam Optimizer (lr=0.1) and selected the best performing one (ADAM) to compensate for convergence speed discrepancies.

RetroGAN outperforms AuxGAN in the majority of similarity benchmarks. In particular, RetroGAN sets the state of the art (SOTA) on the rare-words benchmark (C660). In the similarity results for Full, we note the same observations that were noted in AuxGAN: there are some inconsistent gains and losses, which may be due to the combination of loss functions which may make the systems imprecise; although they spread the knowledge throughout the embeddings, they lose some precision when compared with the original retrofitted embeddings.

[6]Open source at: https://github.com/nmrksic/attract-repel

We wanted to compare the out-of-knowledge (OOK) performance more in depth and to do this, we joined the words in SimLex (SL) and SimVerb (SV) and selected increasingly larger amounts of them ({5,10,25,50,75,100}%). We then selected the constraints that included these words, trained RetroGAN and AuxGAN with these constraints, and evaluated performance on SL, SV, and C660. Part of this can be seen in table 2.4. Evidently, RetroGAN's performance increases every time that new

	5%			10%			25%			50%		
	SL	SV	C660	SL	SV	C660	SL	SV	C660	SL	SV	C660
Attract-Repel	0.347	0.355	0.113	0.550	0.589	0.187	0.701	0.700	0.217	**0.759**	**0.747**	0.252
AuxGAN	0.615	0.510	0.453	0.667	0.569	0.470	0.679	0.581	0.475	0.685	0.600	0.490
RetroGAN	**0.624**	**0.538**	**0.489**	**0.701**	**0.652**	**0.493**	**0.738**	**0.690**	**0.502**	0.755	0.716	**0.511**

Table 2.4: Results for Out-of-Knowledge; best values in **bold font**.

constraints are added, whereas AuxGAN's performance begins to peak after 25% of the constraints which may indicate more efficient knowledge distribution thanks to the cyclic system. Later on, the performance of RetroGAN kept increasing, but was less than the base retrofitted embeddings, possibly because of the lack of precision from the combination of losses. We note that the max-margin loss from (Ponti et al., 2018) is necessary for high performance in all the tests. We also notice that the cyclic (cyclic max-margin and cycle conditional discriminator) losses are essential for improved performance on the OOK and rare-word similarity benchmarks. We also see that the removal of the cyclic max-margin loss speeds up early learning and its addition stabilizes later learning respectively which may indicate a need to balance this. Future work will explore how to balance this losses, but it may be possible to put a scheduler to enable the loss after a peak.

2.4.2 Deep Relationship Discovery (DRD)

Now that RetroGAN provides post-specialized, knowledge-flavored word embeddings for downstream commonsense inference tasks, we shift to the design and development of models for such tasks. In the original work of (Colon-Hernandez et al., 2019), a module named *deep relationship discovery* (DRD) is developed side-by-side with RetroGAN.

Intuitively, DRD first seeks further abstractions of the post-specialized word embeddings for a given concept pair, then concatenates the abstractions so as to reason about the concept pair as a whole, and finally use feed-forward Deep Neural Network models to *discover* the strengths for all enumerated assertions. In sections 2.5 and 2.6, I will demonstrate that it is the strength-indicating scores by DRD that PatchCom benefits the most from.

Architecture of DRD

Figure 2·7 shows the complete architectural design of DRD.

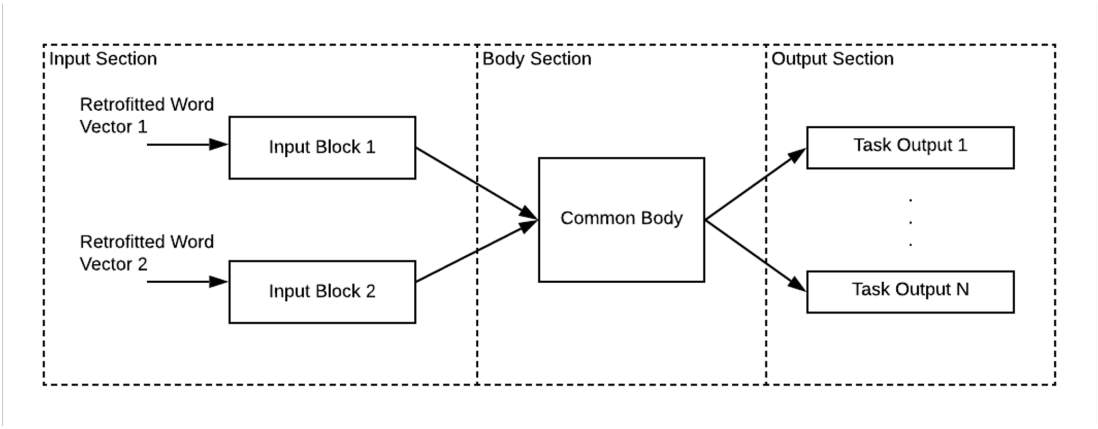


Figure 2·7: Architecture of DRD. Note its three stages of semantic processing, transfer learning, and task specialization.

Preliminary Results of DRD

Figure 2·8 shows a small snapshot of DRD's reasoning ability:

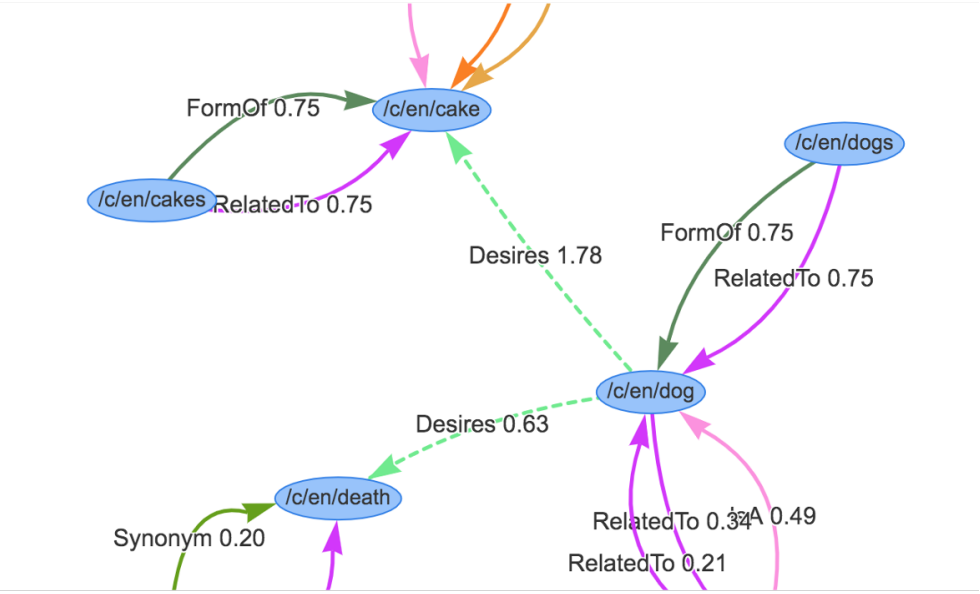


Figure 2·8: Visualization of a very small snapshot of DRD's reasoning outcomes. Note that some of the concepts were not originally connected in ConceptNet, but DRD is able to connect them.

2.5 Disambiguation with RetroGAN-DRD

To recap, in section 2.2 and 2.3, I discussed the task setups for both prepositional phrase attachment and pronoun corefernence, as well as the more naive approach of directly querying for commonsense assertions from ConceptNet (Speer et al., 2017). I then pointed out shortcomings of this naive approach, and suggested that a more advanced neural commonsense inference approach is warranted and might work better. This led me to discussing the sophiscated RetroGAN-DRD system in section 2.4.

In this section, I discuss how PatchComm utilizes RetroGAN-DRD to help it better disambiguate both prepositional phrase attachments and pronoun coreferences. The setups for both tasks are the same as described in sections 2.2 and 2.3. Consequently, all the details on querying RetroGAN-DRD that I describe below apply strictly to pairs of concepts/words, (c_1, c_2).

2.5.1 Post-specialized Embeddings

This part is straightforward: to use DRD downstream, the first order of business is to obtain post-specialized embeddings for $c_!$ and c_2 using RetroGAN. The execution pipeline here is exactly the same as the standalone RetroGAN. In particular, FastText (Bojanowski et al., 2017) is first used to convert both c_1 and c_2 into "plain" word embeddings. Then, these two plain embeddings are fed into RetroGAN and converted into their post-specialized counterparts.

One implementation detail that differs, I note, is that instead of querying with only c_1 and c_2, PatchComm queries RetroGAN with a neighborhood of c_1 and a neighborhood of c_2. The neighborhoods are the same size. To build the neighborhoods, we use the off-the-shelf Faiss library to do cosine-similarity search over ConceptNet NumberBatch (Speer et al., 2017); we set the number of neighbors to be 5. Alternatively, we can do search over FastText for neighboring plain word embeddings,

and then use RetroGAN to post-specialize them. In practice, we opted to search over NumberBatch instead of FastText, because NumberBatch being the retrofitted version of ConceptNet itself should provide better semantics for downstream DRD.

There are two reasons that we query DRD with neighborhoods of c_1 and c_2 instead of c_1 and c_2 alone. The RetroGAN-oriented reason is that we essentially cannot be certain that RetroGAN's output for a concept is in fact the "best" numerical representation for the underlying semantics of that concept, especially in the case of context-independence with PatchComm; consequently, we surround RetroGAN with a small neighborhood of ConceptNet numerical representations, just to be sure. The DRD-oriented reason, which I discuss next, is that DRD is not built to normalize its output scores across all the task models (i.e. one task model per ConceptNet relation), so in order to do basic numerical comparisons across the relations, each relation needs to be normalized with respect to itself.

2.5.2 Querying DRD

To query DRD with c_1's and c_2's neighborhoods, I implemented the following algorithm:

```
Algorithm: DRD Querying
1. Collect k neighbors of c1 and k neighbors of c2.
2. For each relation r (in ConceptNet):
    2.1. Compute DRD scores for all (k+1)^2 pairs
         of matchings.
    2.2. Compute the normalized score of DRD(c1, c2)
         using these scores.  The result will be
         used as the final score for the current
         relation r.
3. Return the max score across all relations.
```

For step 2.2 in the algorithm, in order to find the best way to normalize each

relation's DRD score, I experimented with the following four different normalization techniques:

- Mean:

$$s = \frac{DRD(c_1, c_2)}{\text{mean}(\sum_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y))}$$

- Median:

$$s = \frac{DRD(c_1, c_2)}{\text{median}(\sum_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y))}$$

- Min-Max:

$$s = \frac{DRD(c_1, c_2) - \min_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y))}{\max_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y) - \min_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y)}$$

- Z-score:

$$s = \frac{DRD(c_1, c_2) - \text{mean}(\sum_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y))}{\text{std}(\sum_{(x,y)\in\mathcal{N}(c_1)\times\mathcal{N}(c_2)} DRD(x,y))}$$

From a small set of ablation studies I conducted, I observed that normalization by median works the best.

2.5.3 Experiments and Results

To compare PatchComm with RetroGAN-DRD to PatchComm with ConceptNet, I simply evaluated PatchComm with RetroGAN-DRD on the same experiments as described in sections 2.2 and 2.3, for prepositional phrase attachment and pronoun coreference, respective. Below, I concatenate the results of PatchComm with ConceptNet from sections 2.2 and 2.3 into one table, and append results of PatchComm with RetroGAN-DRD to that table.

	Baseline	Baseline + PatchComm w/ ConceptNet	Baseline + PatchComm w/ RetroGAN-DRD
Self-created	57.0%	67.0%	**80.0**%
WSC273	31.1%	39.9%	**67.4**%

Table 2.5: Baselines are spaCy (top) and NeuralCoref (bottom).

Chapter 3

ProGeneXP

3.1 ProGeneXP Overview

In previous chapters, I discussed my approach to using commonsense to resolve context-independent, sentence-level natural language ambiguities. I advocated for the necessity of commonsense, because it can work well and, more importantly, because it renders the disambiguation process transparent.

In this chapter, I turn my discussion onto ProGeneXP, a general-purpose framework that directly targets at transparency and simultaneously makes it easy to incorporate commonsense reasoning mechanisms for the future. Figure 3·1 shows the overall diagram for ProGeneXP:

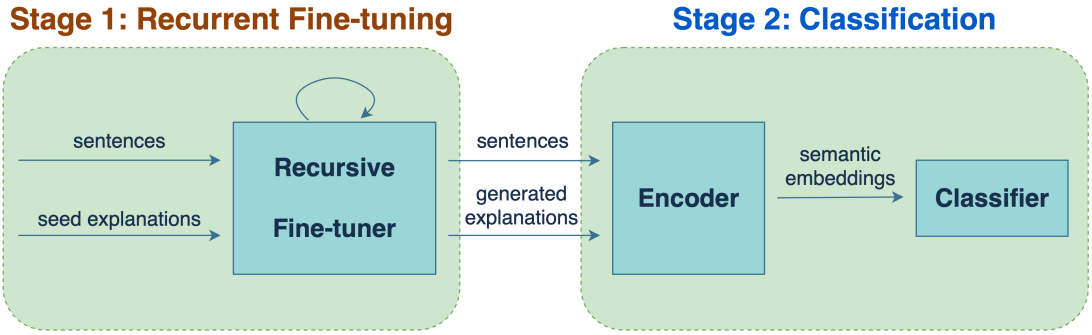


Figure 3·1: Overall diagram sketch for our approach. The self-loop at the top of the Recursive Fine-tuner (RF) indicates the recursive-style learning that we detail in section 2.1.

In particular, ProGeneXP consists of two stages:

- In the first Recurrent Fine-tuning stage, ProGeneXP fine-tunes neural networks to make their implicit semantics transparent by eliciting it as natural language descriptions.

- In the second Task Specialization stage, ProGeneXP evaluates the quality of the descriptions on a set of downstream Pronoun Coreference tasks.

My motivation for the first stage is, of course, to establish AI transparency to whatever extent possible. When it comes to introducing transparency into intrinsically opaque neural networks, my idea is to ask the networks themselves to "translate" their understanding and semantic representations of the world into a form that is most transparent to us humans. My motivation for aiming at the general problem of pronoun coreference for the second stage is, once again, that pronoun coreference ambiguity is perhaps the most significant hurdle to machines understanding natural language. Here, ProGeneXP is in principle not bounded by the same syntactic limitations as we saw of Winograd Schema styled sentences in section 2. My end goal for ProGeneXP is to seek an equilibrium between model transparency from the ground up, and model performance through years of training and evaluation.

For the remainder of this chapter:

- In section 3.2, I discuss the Recurrent Fine-tuning module, as well as my initial probing of the human-understandability of its generated natural language descriptions.

- In section 3.3, I discuss the Task Specialization module with a focus on the specific task of pronounce coreference in Winograd Schema inspired sentences.

- In section 3.4, I discuss how ProGeneXP can be incorporated into PatchComm, to help PatchComm further resolve pronoun coreference ambiguities.

3.2 Recurrent Fine-tuning

In the Recurrent Fine-tuning module, a pre-trained large language model is fed with open-ended pronoun-ambiguous natural language sentences, and then fine-tuned, in a supervised fashion, to generate natural language. descriptions[1] for those sentences.

Despite the word *recurrent* in its name, the Recurrent Fine-tuning (RF) module does not have to incorporate any Recurrent Neural Network (RNN) unit, or the like. The word *recurrent* is, instead, used liberally to mean that a model's later behavior and knowledge depend on the model's earlier behavior and knowledge, respectively. This, in fact, is analogous to an RNN in some abstract way.

To bootstrap the module, a human superviser must initially provide a small set of supervising descriptions. But after this initial supervision, the hope is that the module learns to generate likewise descriptions for sentences provided at later times without supervising descriptions. Of course, at some point, it will become inevitable that the module's behavior dwindles after having undergone a significant period of time without any human supervision. When this happens, a human superviser can once again step in and provide further supervision for the module to "re-bootstrap" or readjust. At the same time, the human can once again observe whether the module will have indeed readjusted its behavior and knowledge expectedly. This establishes a healthy cycle between one or more human supervisers and the module that is much like a classroom setting with a teacher and a student. I strongly believe this form of interactive instruction is a key ingredient to making NLP models transparent, understandable, and friendly.

In short, the two expectations I have from the Recurrent Fine-tuning module are as follows:

[1]Despite the name, these *descriptions* may frequently be thought of as some form of *explanation* or *justification*. In fact, as I will discuss in section 3.3, many of the initially provided human-curated descriptions look very much like if-then rules that help explain or justify why a pronoun should corefer to its correctly corresponding entity instead of any other entity.

- By providing initial supervision and asking the module to analogize upon them, we can hope to obtain transparent expressions of opaque semantics in Deep Neural Network language models. Admittedly, analogy is a longstanding unsolved problem in Artificial Intelligence and is not the focus of this dissertation. My effort here is but one simple dabble that perhaps deserves a more analogy-focused research discussion in the future.

- So that it is of utility to have a human provide descriptions via interacting with the module at the human's convenience, the module must be able to learn continually. From this perspective, the Recurrent Fine-tuning module can be regarded as an instantiation of the idea of *continual learning* from the sub-area of Machine Learning called Meta Learning (Finn et al., 2017; Javed and White, 2019; Beaulieu et al., 2020).

In addition to attempting to meet the two expectations above, the Recurrent Fine-tuning module also borrows inspiration from the work of (Shwartz et al., 2020), which proposes an approach to question answering by literally setting large language models to talk to themselves. I personally speculate the reason this works is because large-scale Transformer (Vaswani et al., 2017) based language models are capable of readjusting their internal "attention spans" simply by constantly talking to themselves but in complex ways, thanks to the very nature of attention mechanism (Bahdanau et al., 2015).

To set the stage for my discussions in sections 3.3 and 3.4, the Recurrent Fine-tuning module is currently set to generate descriptions for pronoun-ambiguous natural language sentences. My motivation here is practical: I want to seek a better description-finding mechanism for PatchComm's existing pronoun coreference mechanism. I have two reasons for doing this — or rather, PatchComm's existing description-finding mechanism is flawed in two ways — as follows:

- Recall in chapter 2 that PatchComm's pronoun coreference mechanism suffers from its poor ability to extract so-called *description tokens* from sentences. It is difficult, if not impossible, to always be able to find one or a group of tokens within a sentence that captures all the relevant semantics from that sentence.

- Frequently, much of the knowledge required for coreference are outside of a sentence and must come from a more global context. Therefore, looking for optimal descriptions within a sentence may not work.

Consequently, it is natural that I think about applying ProGeneXP's ability to generate natural language descriptions to PatchComm, in place of PatchComm's rigid description tokens.

In addition, due to the natural language characteristic, the syntactic constraints as stipulated in section 2.3 does not immediately apply to ProGeneXP. Of course, at some later stage for task specialization on Winograd Schema sentences, those constraints can be added back.

3.2.1 Implementation Details

Algorithmically, after the initial supervision, every time it is fed new sentences, the Recurrent Fine-tuning module utilizes its most-recently acquired knowledge to generate corresponding descriptions for those inputs. Then, RF uses these new sentence-description pairs as new data to update its ability to self-generate explanations.

Because pre-trained large language models capture distributional semantics over an enormous amount of natural language texts, they certainly have at least some of the knowledge that are outside of particular sentences yet necessary for disambiguating the pronouns within those sentences. In addition, language models naturally integrate, at least to some degree, the specific semantics from a given sentence with the distributional semantics from all the corpora they have seen.

In its ultimate version, the Recurrent Fine-tuning module runs in a never-ending fashion, both learning and generation. In practice, however, checkpoints have been administered for convenience and for experimental purposes.

To provide initial human supervision, I selected 224 sentences from the Definite Pronoun Resolution (DPR) dataset (Rahman and Ng, 2012) and manually wrote an English explanation for each of these sentences. After that, I designated the following three checkpoints:

- To reach the first checkpoint, I fine-tuned, for the first time, HuggingFace's off-the-shelf T5-Large (Raffel et al., 2019) based text summarizer training pipeline[2] on these intial 224 pairs of (sentence, description).

- To reach the second checkpoint, I first used the previously fine-tuned text summarizer to generate descriptions for all the remaining DPR sentences that were excluded from the initial supervision. I then fine-tuned, for the second time, the summarizer on all $1,886$ of DPR sentence-description pairs. For the sentences that already appeared in the initial supervision, I just used my curated descriptions, but for the ones that did not appear, I used the newly generated descriptions.

- To reach the third and final checkpoint, I first used the previously fine-tuned summarizer to generate descriptions for all $9,248$ WinoGrande-debiased training sentences and $1,267$ WinoGrande development sentences (Sakaguchi et al., 2020), where the blank of each sentence is filled by the correct candidate. I then fine-tuned this summarizer, for the third time, on all of WinoGrande sentence-description pairs. This time, all descriptions are generated.

[2]Code base is publicly available at `https://github.com/huggingface/transformers/tree/main/examples/pytorch/summarization`.

The end result is a T5-based text summarizer that has been recurrently fine-tuned for three times.

3.2.2 Probing Examples

Having obtained this recurrently fine-tuned summarizer, next I investigate the quality of its generated descriptions using some probing examples.

To showcase its transparency and its ability to learn upon initial supervising sentences and analogize to new sentences, I probe the Recurrent Fine-tuning module with natural language inputs and present some exemplary generated descriptions.

When probing, I look for two qualities:

- Without taking context into account, RF's generated natural language descriptions should abide by straightforward commonsense. Indeed, if a generated description is simply nonsense, then there is no need to further take context into account.

- Taking context into account, such a generated description should "appear" relevant to the sentence from which the description is derived. Indeed, much of human interpretation of things relies on the appearance of things — any attempt at formulating the notion of *interpretability* mathematically begs the question of how interpretable those formulations, in turn, are.

In addition, I stick to the following suggestions for writing the initial supervising descriptions:

- For each pronoun-ambiguous sentence, an explanation must be one-sentence long, in English, and must elicit how the user resolve the pronoun of interest. For example, to resolve that "it" in "The bee landed on the flower because it had/wanted pollen," a user might write "Bees like pollen/Flowers have pollen" as explanation, eliciting the background information of utility.

- An explanation needs to hint at accepting the correct answer, or rejecting the wrong answer, or both. For example, in the previous example twin sentences that differ in only "had/wanted," a plausible explanation is "Bees like pollen *and* flowers have pollen," collecting both information into one sentence.

For investigation, we tried batch sizes of 4 and 8 for each fine-tuning checkpoint, but did not observe significant differences between the generated explanations by the batch-size-4 models and those by batch-size-8. In addition, we observed that RF model sizes increase dramatically after each checkpoint. Table 3.1 collects the sizes of all RF models.

1st checkpoint (224)	batch= 4	2.8G
	batch= 8	2.8G
2nd checkpoint (1,886)	batch= 4	20G
	batch= 8	11G
3rd checkpoint (10,515)	batch= 4	127G
	batch= 8	127G

Table 3.1: RF model sizes. Note that for the 3rd checkpoint, $10,515 = 9,248$ (WinoGrande-train-debiased) $+1,267$ (WinoGrande-dev).

Evidently, model size becomes an issue as the module is continually fine-tuned. I aim to address this issue in my future work.

Table 3.2 presents some of the examples I used, in comparison with their corresponding human inputs if applicable, as well as with outputs of "plain" models who did not go through the recurrent fine-tuning process.

The infection spread throughout the building because [it] was airborne.	
Human	Airborne viruses can spread quickly in ventilated in-door areas.
Plain	The infection spread throughout the building because it was airborne.
RF	Airborne viruses can spread quickly in a building.

The infection spread throughout the building because [it] was ventilated.	
Human	Airborne viruses can spread quickly in ventilated in-door areas.
Plain	Infection spread because the building was not ventilated.
RF	Ventilation can spread infection throughout a building.

The smoke spread through the building because [it] was caught on fire.	
Human	—
Plain	Smoke spread through the building because it was on fire.
RF	Smoke is created when a building is on fire.

The smoke spread through the building because [it] came from the fire.	
Human	—
Plain	Smoke spread through the building because it came from the fire.
RF	Smoke from a fire can spread quickly in a building.

Table 3.2: Probing examples for the Recurrent Fine-tuning module. At the top, the sentences are taken from the DPR dataset for initial supervision with human-curated descriptions. At the bottom, the sentences are provided at probing time by us. **Plain** means the default T5-for-Conditional-Generation text summarizer taken directly from the HuggingFace library without fine-tuning of any kind. **RF** stands for our Recurrent Fine-tuning approach. Also note that the "[]" in the sentences are added to the table to clarify the pronouns of interest; they do not appear in the datasets.

These results clearly suggest the summarizer's ability to analogize and produce descriptions that are consistent with what we want to expect of such models. Namely, in both twin sentence pairs, we see that the plain model simply regurgitates the original sentences, lacking efforts in resolving the ambiguous pronouns and confusing the sentence semantics with their negative counterparts. The recurrently fine-tuned model, on the other hand, learns to associate the correct entities with their local contexts in the initial sentences and carries this ability over to new sentences.

3.3 Task Specialization

In Artificial Intelligence, things that look good to people frequently do not work well for machines. Consequently, to see whether the human-understandable outputs by the Recurrent Fine-tuning (RF) module help NLP models perform better, in this section I discuss how I evaluate the quality of generated descriptions by Recurrent Fine-tuning. I have chosen the task of pronoun coreference, as discussed before.

More specifically, I set up pronoun coreference here as a classification problem, where the ambiguous pronoun is to be correctly classified into one of the candidates. Inspired by the work of WinoGrande (Sakaguchi et al., 2020), I have utilized a very similar method, where, for every sentence, its ambiguous pronoun is replaced by the correct candidate, and all tokens after the pronoun are delimited and treated them as contextual information for the encoder and classifier. Naturally in the Winograd Schema style setting, pronoun coreference is set up as binary classification because there are only two candidates per sentence.

As shown in Figure 3·1, the Task Specialization module consists of an encoder followed by a classifier. This module is kept as simple as possible for now, so that I can focus on investigating the effectiveness of the Recurrent Fine-tuning module.

3.3.1 Implementation Details

For the encoder, I utilized off-the-shelf pre-trained BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) models from the HuggingFace library[3]. For the pre-trained BERT models, in turn, I utilized both the *cased* and *uncased* models, which means that the BERT model was pre-trained on cased and uncased English texts, respectively. In addition, for the fine-tuning data, I utilized two different version of the WinoGrande debiased training set of $9,248$ sentences: with RF-generated descriptions

[3]All pre-trained Transformer-based models, such as BERT and RoBERTa, are publicly available at `https://github.com/huggingface/transformers`.

appended to each sentence, and without. Therefore, in total, I implemented the encoder in six different versions:

- BERT cased, fine-tuned with descriptions
- BERT cased, fine-tuned without descriptions
- BERT uncased, fine-tuned with descriptions
- BERT uncased, fine-tuned without descriptions
- RoBERTa, fine-tuned with descriptions
- RoBERTa, fine-tuned without descriptions

For the classifier, I simply utilized PyTorch's built-in linear layer with back-propagation turned on to allow for parameter tuning. The input size for the classifier is 768, which is the embedding size of both BERT and RoBERTa. The output size is 2, due to binary classification.

During each epoch, the training is with validation, with a $80\% - 20\%$ split on the said WinoGrande debiased training set, whether the dataset comes with descriptions or not. We trained the module for a total of 20 epochs with batch size 32.

After training, the module is tested on the WinoGrande validation set of $1,267$ sentences. I selected the validation set instead of the test set, because the test set does not have labels. Recall that the Recurrent Fine-tuning module has been used to also generate descriptions for the WinoGrande validation set. Thus, we tested the Task Specialization module on the validation set twice, with and without descriptions.

3.3.2 Preliminary Results

Table 3.3 shows all the results of the Task Specialization stage. For baselines, we included both the coin-flip probability of 50% as well as the WinoGrande baseline presented in (Sakaguchi et al., 2020).

Test \ Train	BERT (cased)		BERT (uncased)		RoBERTa	
	–	with descs	–	with descs	–	with descs
Baseline (coin flip)	0.5					
Baseline (WG)	–	–	0.658	0.649	0.793	0.791
WG-valid	0.867	0.859	0.863	0.852	0.855	0.856
WG-valid with descs	0.840	**0.867**	0.828	**0.858**	0.844	**0.862**

Table 3.3: Test results for all 6 fine-tuning settings tested on 2 different test settings. "WG" stands for WinoGrande. The WinoGrande baselines are taken directly from Table 3 of (Sakaguchi et al., 2020).

Albeit simple, the ProGeneXP approach outperforms the baseline WinoGrande approaches across the board, suggesting that the Recurrent Fine-tuning approach provides useful descriptions that help the Task Specialization module to hone in on relevant semantics.

I also observe that my results, despite higher accuracy across the board, also have very high precision with each other. I suspect this is due to the module successfully exploring some underlying statistical bias in the WinoGrande debiased dataset that may not be interesting or easily made transparent to people.

Finally, I note that for the RoBERTa results, training and testing the encoder-classifier stack with descriptions show promise in outperforming trainind and testing without descriptions. For future work, I suspect that more training loops can help further distinguish the performance of various models. I also look forward to incorporating neural commonsense inference mechanisms, like the ones discussed in chapter 2, into ProGeneXP for better performance on disambiguation at large.

3.4 PatchComm with ProGeneXP

In this section, I draw out a small amount of implementation details of an immediate future work where ProGeneXP is incorporated into PatchComm (chapter 2) to help PatchComm find better semantic descriptions for disambiguation tasks.

Recall in sections 3.2 and 3.3 I discuss that PatchComm's current pronoun coreference approach suffers from the rigid notion of *description tokens* from within the sentences. I explain the two reasons why description tokens are sub-optimal:

- It is frequently too limiting to find only one token, or a group of tokens, within a sentence that captures all the relevant semantics for understanding the coreference of a pronoun within that sentence.

- It is frequently the case that the necessary knowledge for coreference may be outside of a sentence, so looking for a good description inside a sentence would not work.

Basically, I take the entire pipeline as shown in Figure 3·1 except for the final classifier. When dealing with a pronoun-ambiguous sentence, ProGeneXP is first invoked to generate a description for that sentence and to encode a concatenation of the original sentence with the description. The result is a 768-dimensional embedding, because the encoder is implemented with BERT and/or RoBERTa. After that, I perform dimensionality reduction on the sentence-description embedding to obtain a 300-dimensional embedding. This then becomes the description vector (instead of token) for PatchComm to align with candidate entities in pronoun coreference. Because everything is in vector form, the alignment is done by RetroGAN-DRD (Colon-Hernandez et al., 2019).

Results from initial experiments are to follow after this dissertation.

Chapter 4

DialComm

4.1 DialComm: Discourse Understanding By Commonsense

In this speculative chapter, I discuss DialComm. DialComm is a general-purpose framework that aims at understanding discourses by building semantic representations for discourses. To build robust discourse representations, it is important to combine context-independent sentence-level disambiguation and commonsense inference, and tackle context-dependent ambiguities that require discourse-level semantics to resolve. As circular as this may sound, discourse-level semantic representations and disambiguation mechanisms aid each other, often in loops.

Understanding natural language discourse in robust ways is a longstanding research challenge; for example, see (Scha et al., 1986). Presently, DialComm is built upon PatchComm's ability to handle sentence-level ambiguities, inspired by the Frame semantic representation (Minsky, 1974) and research in interactive natural language programming such as Metafor (Liu and Lieberman, 2005), and fueled by commonsense inference mechanisms.

In its current version, DialComm consists of a Sentence Representation Builder (SRB) that directly utilizes PatchComm's (Chapter 2) capability of using commonsense knowledge and reasoning to disambiguate sentences, and a Discourse Representation Builder (DRB) that recursively draws on information collected from previous sentences so as to understand new sentences within the established context. Figure 4·1 shows the architecture of DialComm.

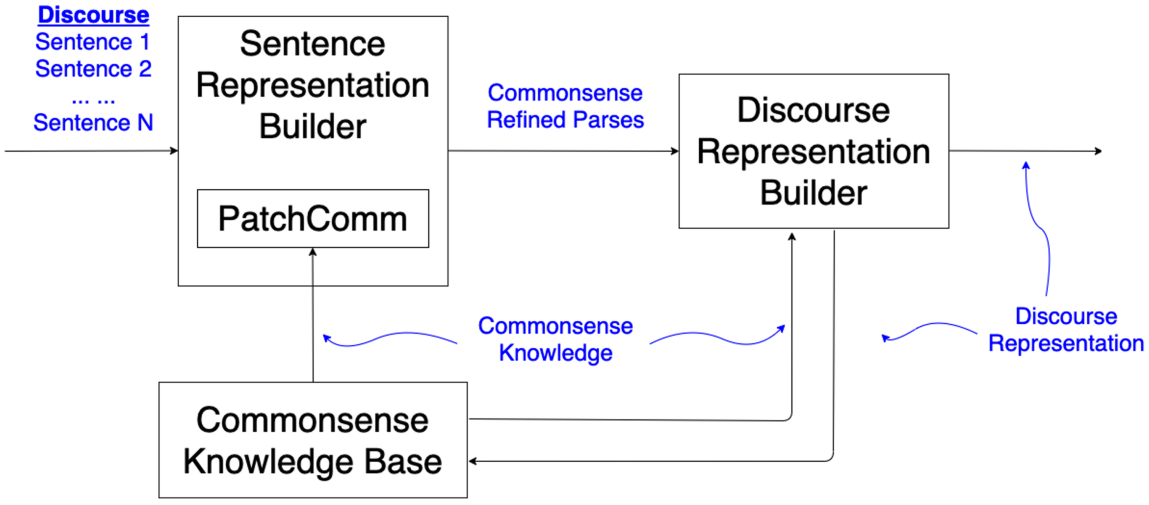


Figure 4·1: Architecture of DialComm.

In order to achieve this, there must be a unifying representation that congregates, and allows for the updating of, information across all sentences. We have chosen to use the frame data structure (Minsky, 1974) to implement this unifying discourse-level representation.

One natural by-product that can be obtained from this frame representation, is sentence-level pronoun coreference disambiguation. In the next section, I demonstrate DialComm's ability to apply discourse-level representation back to sentence-level ambiguous pronouns. Although this pronoun coreference disambiguation capability is only part of downstream capabilities such as question answering and story summarization, the ability to reflect on one's previous behaviors and outcomes and explain why decisions are made, is necessary for transparent intelligent agents.

4.2 Details of DialComm

In this section, I demonstrate DialComm's story understanding and commonsense inference mechanisms, by walking through a concrete example story. We will see that pronoun coreference disambiguation is a natural by-product of this story understanding process. The story is inspired by the exemplary *Birthday Party* scenario that has been extensively studied in Artificial Intelligence — for example, see (Clemenson, 1977). My version of the Birthday Party story consists of the following sentences:

S1: Robbie and Susie are going to Marvin's birthday party.
S2: **One** of **them** wants to buy a kite.
S3: "But **he** already has **one**," **he** says, "**he** will make **you** take **it** back."

Figure 4·2: A simple but pronoun-ambiguous Birthday Party story.

At its core, DialComm follows an incremental approach that is implemented as a recursive algorithm. The central Frame representation is updated whenever new information is detected in a new sentence. When DialComm reads a sentence, it uses PatchComm (Xin et al., 2021) (essentially spaCy (Honnibal et al., 2020)) to parse that sentence into parts of speech and syntactically connected phrases, and then attempts to match these extracted information against its own knowledge that has been pre-built-in for bootstrapping.

At the top level, DialComm understands that there are three root entities — the Person, Thing, and Event entities — but does not include these labels into any frame it builds. After reading S1 of Figure 4·2, PatchComm extracts the key information of *Robbie and Susie*, *Marvin*, *Marvin's birthday party*, and *going*. Then, DialComm uses its built-in knowledge to know that:

- Marvin's birthday party is an event, with Marvin being the host.
- If Person X goes to Person Y's birthday party, then Person X is a guest. So Robbie and Susie are the guests to Marvin's birthday party.
- Host and guests are all Persons.
- Robbie is male, Susie is female, and Marvin is male; their pronouns should be assigned accordingly[1].

Putting these together, DialComm builds the following initial frame:

Hosts	**Marvin**	**Pronoun: He**
Guests	**Robbie**	**Pronoun: He**
	Susie	**Pronoun: She**

Table 4.1: Frame after reading S1. Note that the frame includes only relevant information. For example, the frame does not repeat the fact that the event is birthday party, because this is an event frame and the event is already known. Putting spotlight over only relevant information, is central to Minskyian frames (Minsky, 1974).

After reading S2 of Figure 4·2, PatchComm takes advantage of its pronoun coreference ability to guess some preliminary results that DialComm does not yet rely on; also, PatchComm extracts the entire phrase of *want to buy a kite*. Then, DialComm uses its built-in knowledge to know that

- Birthday party guests usually buy gifts for birthday party hosts.
- *Want to* means not yet done, so a discussion might ensue and be anticipated.

With these information, DialComm updates the frame, which becomes: DialComm resolves that "them" refers to Robbie and Susie, because:

[1]Note that this stipulation is so that DialComm can begin doing any processing at all. I decline all political arguments on pronoun and gender identity.

Hosts	Marvin	Pronoun: He
Guests	Robbie	Pronoun: He
	Susie	Pronoun: She
Gifts	**Kite**	**From: Robbie** **To: Marvin** **Status: Maybe**
		From: Susie **To: Marvin** **Status: Maybe**

Table 4.2: Frame after reading S2. *From* and *To* capture the fact that a gift needs a giver and a receiver. *Status: Maybe* captures the fact the uncertainty of whether the gift will indeed be bought. The inclusion of both *From: Robbie* and *From: Susie* is because DialComm resolves that "them" refers to Robbie and Susie, but needs more discourse-level context to resolve "one."

- From before, when DialComm sees *want to* and anticipates a discussion on gifts between guests, who DialComm assumes are the gift buyers.

- DialComm confirms that the guests are Robbie and Susie, by referencing the existing frame from the previous step. By default, DialComm assumes that plural pronouns such as "them" refer to entities that have the most in common, prioritizing the most specific commonalities and incrementally moving to more general commonalities. In the example, Robbie and Susie are both guests whereas Marvin is a host, even though all three are Persons. Therefore, DialComm confirms that "them" refers to Robbie and Susie.

Finally, after reading S3 of Figure 4·2, DialComm immediately knows that the dialogue is between Robbie and Susie, because from before DialComm has been anticipating a discussion on gifts between them. Then from "he says" DialComm knows that Robbie is the speaker, by referencing the existing frame for pronoun information. This leads to the final version of the event frame for the story: Having this final frame,

Hosts	Marvin	Pronoun: He
Guests	Robbie	Pronoun: He
		Dialogues: **"But he already has one"** **"he will make you take it back"**
	Susie	Pronoun: She
		Dialogues:
Gifts	Kite	From: Robbie To: Marvin Status: Maybe
		From: Susie To: Marvin Status: Maybe

Table 4.3: Final frame after reading S3.

DialComm can revisit all the unresolved pronouns, starting with the current sentence (S3) and then doing a full pass through the story again. For Robbie's dialogues in S3, DialComm has built-in linguistic knowledge to know that, within quoted utterances, *I/me/my* self-refers to the speaker; *you/your* refers to whomever the speaker is speaking with; and third-party pronouns like *he/him/his*, *she/her/her*, *it/it/its*, and *they/them/their* refer to whoever outside of the dialogue that go by those pronouns accordingly. Using such knowledge, DialComm easily infers that the *he* in Robbie's dialogue refers to Marvin, *you* refers to Susie, and *one* and *it* refers to Kite.

In its ultimate version, DialComm should further infer that Susie will thus decide not to buy a kite for Marvin's birthday party. However, in its current version, DialComm still needs to leverage much more sophisticated commonsense knowledge and inference mechanisms to draw such conclusions.

4.3 Application to Programming in Natural Language

In order to generalize beyond story-specific mechanisms, below I describe several substrates of the Sentence Representation Builder (SRB) of DialComm that attempt to reliably extract information from sentences. Then, I describe how these mechanisms are useful for a downstream task of programming in natural language. All implementations are grounded in a combination of dependency parsing mechanisms from PatchComm and commonsense knowledge from ConceptNet[2] (Speer et al., 2017).

For demonstration, I walk through an example story that is directly inspired by the *Bartender* story in (Liu and Lieberman, 2005):

S1: I found a bar with a bartender who was making fancy drinks.
S2: If a drink was in the menu, the bartender would make it.
S3: One customer was rude, so the bartender threw away their drink.

Figure 4·3: A variation of Bartender story.

4.3.1 Substrates for Sentence Representation Builder

Entity Getter

Entity is the most important aspect of DialComm, because all of DialComm's frame semantic information is indexed by entities. I define an entity to mean noun (including proper noun), noun phrase (including proper noun phrase), or any combination of the two. For example, piano (noun), Yamaha (proper noun), subway station (noun phrase), New York City (proper noun phrase), and New York City subway station (combination) are all examples of entity.

In the current version, Entity Getter is able to extract such example entities as above. However, a major challenge is to robustly distinguish single entities that carry

[2]An immediate thought is to use RetroGAN-DRD in place of ConceptNet. However, the current version of RetroGAN-DRD is not as precise as ConceptNet. For DialComm-styled discourse analysis, precision is crucial in applying knowledge.

conjunctives in their names, e.g. *rum and coke* or *peanut butter and jelly sandwich*. With the help of ConceptNet and other forms of knowledge bases, DialComm can identify some of these conjunctives. Still, for a general solution, I suspect that interactive instructions from users will be necessary.

Sub-entity Getter

Sub-entity is a rather loose notion, and by it I mean an entity that roughly belongs to or is contained in another entity. For example in the bartender story in Figure 4·3, a bartender is a sub-entity of a bar, because bartenders can be found at bars.

Using ConceptNet's relations `HasA` and `AtLocation`, I define that an entity E_1 is a sub-entity of another entity E_2, if either E_2 `HasA` E_1 is true or E_1 `AtLocation` E_2 is true. For bartender and bar, ConceptNet has an assertion that says "bartender `AtLocation` bar," and DialComm uses this assertion.

Property Getter

I simply define that an entity has a certain property P, if P is either the adjectival modifier or adjectival complement of the entity. PatchComm provides such syntactic information.

Action Getter

In DialComm, all *actions* are defined and indexed by their corresponding entities. For implementation, I define an action of an entity to be a verb that is either directly connected to the entity (e.g. *the bartender makes the drink*), or indirectly connected to the entity via one or multiple conjunct links *the bartender makes the drink and gives it to the customer*.

Within the larger frame indexed by entities, an action itself is also represented as a frame, where the slots are *Direct Objects*, *Indirect Objects*, and *Conditions*. For

example, in S2 and S3 of Figure 4·3, DialComm knows that *drink* is a direct object of the actions *make* and *throw*, respectively. On top of that, *make* has a condition, `if drink in menu`, which DialComm gathers from the *if*-led adverbial clause.

Inheritance Getter

Last but not least, whenever there are multiple entities and some are more general/specific than others, there must be a way of resolving their relations. In DialComm, ConceptNet's `IsA` relation is used to help determine such *inheritance* relations. In particular, I devise an algorithm for recognizing chained inheritances. For example, suppose we are given the following list of entities,

$$[piano,\ Yamaha,\ instrument,\ Steinway]$$

and would like to order their inheritance relations based on our commonsense understandings about these entities. The inheritance recognition algorithm first establishes a complete list of all `IsA` relations from this list, which looks like

$$[(piano\ \texttt{IsA}\ instrument),$$
$$(Yamaha\ \texttt{IsA}\ piano),$$
$$(Yamaha\ \texttt{IsA}\ instrument),$$
$$(Steinway\ \texttt{IsA}\ piano),$$
$$(Steinway\ \texttt{IsA}\ instrument)]$$

Then, the algorithm observes that (*Yamaha* `IsA` *instrument*) is a chained inheritance, because (*Yamaha* `IsA` *piano*) and (*piano* `IsA` *instrument*) suggest that *piano* is a middle entity. Therefore, the algorithm removes (*Yamaha* `IsA` *instrument*) from the list. Similarly, the algorithm removes (*Steinway* `IsA` *instrument*) from the list, yielding a list of 1-hop inheritances[3]:

[(*piano* `IsA` *instrument*),
(*Yamaha* `IsA` *piano*),
(*Steinway* `IsA` *piano*)]

Although the working *Bartender* example does not make use of the Inheritance Getter, it is a necessity for building larger and more complex frames.

4.3.2 Program Rendering

Just like in section 4.2 for pronoun coreference disambiguation alone, the programming application requires the same parsing and frame-building stages. In addition, there is one more program rendering stage.

In the parsing stage, DialComm invokes PatchComm to produce the following parse trees, with disambiguated prepositional phrase attachments and pronoun coreferences highlighted, just like in chapter 2:

[3]Here, "1-hop" means based on only the knowledge we have. If new knowledge comes up that suggests additional middle entities, the algorithm modifies the list of inheritances accordingly.

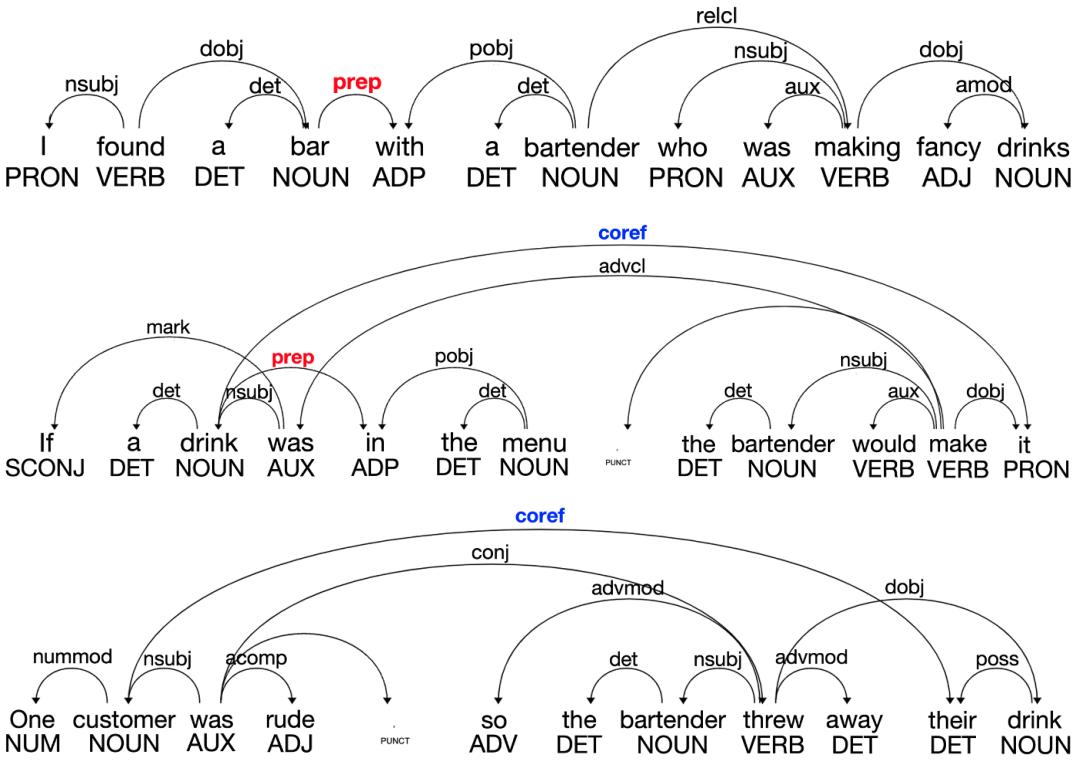


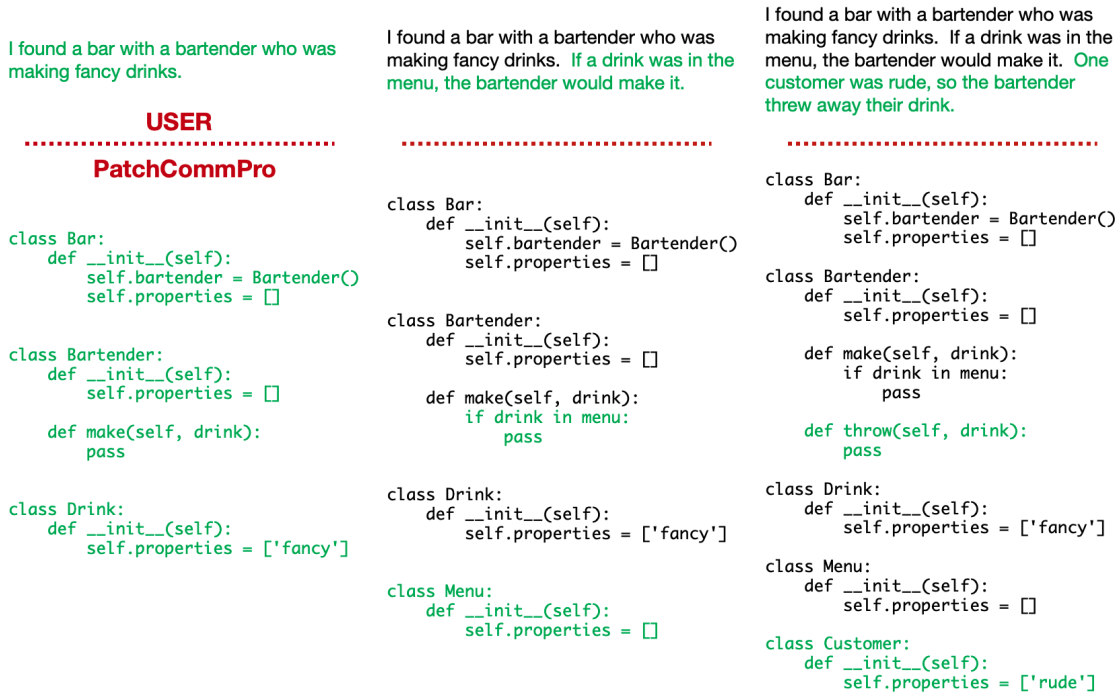
Figure 4·4: Parses for bartender story sentences, highlighting key information.

Then, DialComm invokes the same incremental frame-constructing mechanisms to build and update intermediate frames using the parse trees above. Below, I only show the final version of that frame:

Entities	Actions		Sub-Entities	Properties
Bar			bartender	
Bartender	make(drink)	condition: if drink in menu		
	throw(drink)			
Drink				fancy
Menu				
Customer				rude

Table 4.4: DialComm fills the frame slots as much possible. A sparse frame suggests that more information can be introduced with ease.

And finally, Figure 4·5 shows DialComm's incremental program rendering of all intermediate frames and the final frame (Figure 4.4) into Python-like scripts.



(a) After Table 3.4. **(b)** After Table 3.5. **(c)** After Table 3.6.

Figure 4·5: PatchCommPro's Renderer renders the discourse representation so-far into Python code, in a real-time fashion. Specifically, the green code in (a) is generated after Table 3.4 is generated; green code in (b) after Table 3.5; and green code in (c) after Table 3.6.

When mapping frame semantics to Python-like scripts, DialComm simply follows these rules:

- Entities become Python classes.
- Sub-entities become class instances of their corresponding entities.
- Actions become instance methods of their corresponding entities.
- Properties are simply stored as a list, accessible by class instances.

In addition, thought not shown in the bartender example, inheritances become super-classes and sub-classes.

Also, note that DialComm sticks to Pythonic syntax as much as possible. For example, all class names are capitalized (and camel-cased, though not shown here); every class begins with the `__init__` block, which isn't always required, but is common practice; each class has the *self* keywords where they need to be; and methods are *passed* whenever their procedural details are insufficiently or not at all described.

4.4 Motivations and Future Work for Natural Language Programming

To thoroughly establish the kind of human-to-AI communications depicted in Figure 1.2, outputs of NLU must be converted into computer code written in some high-level, human-readable programming language (PL).

In (Lieberman and Liu, 2006) and (Mihalcea et al., 2006), two paradigms of NL Programming are identified: the *Descriptive* paradigm and the *Procedural* paradigm. It is first and foremost important to note that these paradigms should be working with, not against, each other.

The Descriptive paradigm treats *programming* as the process where humans try to establish discourses or exchange thoughts with computers; for example, a human user might use English to describe the layout of their kitchen to a computer, and the computer internalizes this discourse into a Python program in which the various objects and object-relations in the kitchen are modeled with Objected-Oriented-Programming (OOP) mechanisms. The Procedural paradigm might be considered as the more "conventional" type of NL Programming, which treats programming as the process where humans issue verbal commands in NL to computers without having to translate them into PL. For example, suppose we want to write a Python program that enumerates all integers divisible by 113 between 1 and 10000. If we know how to program in Python, we might write the following program:

```
all_integers = []
for i in range(1, 10000):
    if i % 113 == 0:
        all_integers.append(i)
```

For a human user who does not program in Python or at all, Procedural NL Programming allows them to simply tell their computer to "write a Python program that enumerates all integers divisible by 113 between 1 and 10000."

Genesis, in addition to its ambition in NLU and modeling human intelligence, has a self-aware problem solving process (Winston, 2018) that, in a deep sense, sets up yet another ambitious framework for NL Programming. In that problem-solving process, Genesis first internalizes problems presented in the form of carefully written, simple English stories into Innerese representations (see section 2.2), and then decomposes these representations into substrates known as Just-Do-It programs, which are low-level, directly executable programs. Although strictly speaking, Genesis is not an implementation of the Descriptive NL Programming paradigm, it is a demonstration of how to generate descriptive representations of language and ground them in Procedural implementations. And fundamentally, this is how I believe the Descriptive and Procedural paradigms should work with each other: the Descriptive paradigm constructs the skeleton and the Procedural paradigm fills the skeleton with flesh.

4.4.1 Descriptive and Procedural Programming

The pioneering work in Descriptive NL Programming is Metafor (Liu and Lieberman, 2005), which contributed to the aforementioned works of (Lieberman and Liu, 2006) and (Mihalcea et al., 2006), but unfortunately stopped short. In its final version to date, Metafor consists of, among others, a backend, commonsense-integrated NLP engine that does syntactic parsing and maps parsed outputs to representations of semantic information and commonsense knowledge (e.g. entities, entity relations, actions, properties, etc.); a backend code renderer that renders semantic representations into Python code (at the time of Metafor, Python was in its versions 2.4 and 2.5); and crucially, a frontend User Interface (UI) where human users/programmers interact with the backends by narrating stories sentence-by-sentence. Every time a new sentence is provided in the UI, Metafor looks at the semantic representations already built and the code already rendered, extracts new information (if any) from the new sentence, and updates the existing semantic representation and code with

those new information. Figure 4·6 shows an example of Metafor:

Figure 4·6: An example of Metafor that shows both frontend and backend capabilities. Lower-left corner: the narrative being entered; upper-left corner: an interaction log; upper-right corner: Metafor's representation of the parse tree (for advanced users only); and lower-right corner: the rendered Python code.

A very similar line of work to Procedural NL Programming is *Semantic Parsing*. Unlike Syntactic parsing which maps NL texts onto syntactic parse trees (see

section 2.2), Semantic Parsing maps NL texts onto logical forms. The denotations (i.e. meanings) of such logical forms are identical to the results that would be obtained if these logical forms were implemented in some programming language and then executed. An exemplary semantic parser is SEMPRE[4] (Semantic Parsing with Execution) (Berant et al., 2013; Berant and Liang, 2014), implemented in Java. As an example, suppose we ask SEMPRE, "What is three plus four times two?" SEMPRE understands this ambiguous question could be interpreted as either *three plus four* before *times two* or *four times two* before *three plus*, therefore produces so-called *derivations* that represent the two logical forms below, to capture this ambiguity:

```
(execute (call * (call +
                       (number 3)
                       (number 4)))
                 (number 2))
```

(a) First interpretation of *three plus four* before *times two*.

```
(execute (call + (number 3)
                 (call *
                       (number 4)
                       (number 2))))
```

(b) Second interpretation of *four times two* before *three plus*.

Afterwards, SEMPRE executes both derivations to yield `(number 14)` and `(number 11)`, respectively.

[4]Open source at https://github.com/percyliang/sempre

On the surface, the difference between Semantic Parsing and Procedural NL Programming seems to only be that one produces logical forms as output and the other produces code as output, respectively. But on a deeper level, representing natural language in logic has had a long history of problems and criticisms; perhaps the most quoted criticism is: Logic is too rigid for natural language. True, programming languages are also rigid, but they are more flexible than logic. Therefore, I am more inclined to Procedural NL Programming than to Semantic Parsing.

Directly targeted at the task of Procedural NL Programming is the latest work by OpenAI, Codex[5] (Chen et al., 2021). Building on the GPT family of language models(Radford et al., 2018; Radford et al., 2019; Brown et al., 2020), Codex is specifically trained for generating code. Figure 4·8 provides some examples of Codex at work.

Codex has been shown successful in many application cases, and is now the backbone of GitHub Copilot[6]. But evidently, there are test cases that should be trivial to human programmers but can be used to trick Codex. This hints at the lack of generalizability in Codex and systems like Codex, but the hope is that, by integrating Descriptive capabilities with this kind of SOTA Procedural capabilities, we will make progress in realizing the dream of programming machines in natural languages and thereby establishing thorough human-to-AI communications.

[5]See https://openai.com/blog/openai-codex/
[6]See https://copilot.github.com/

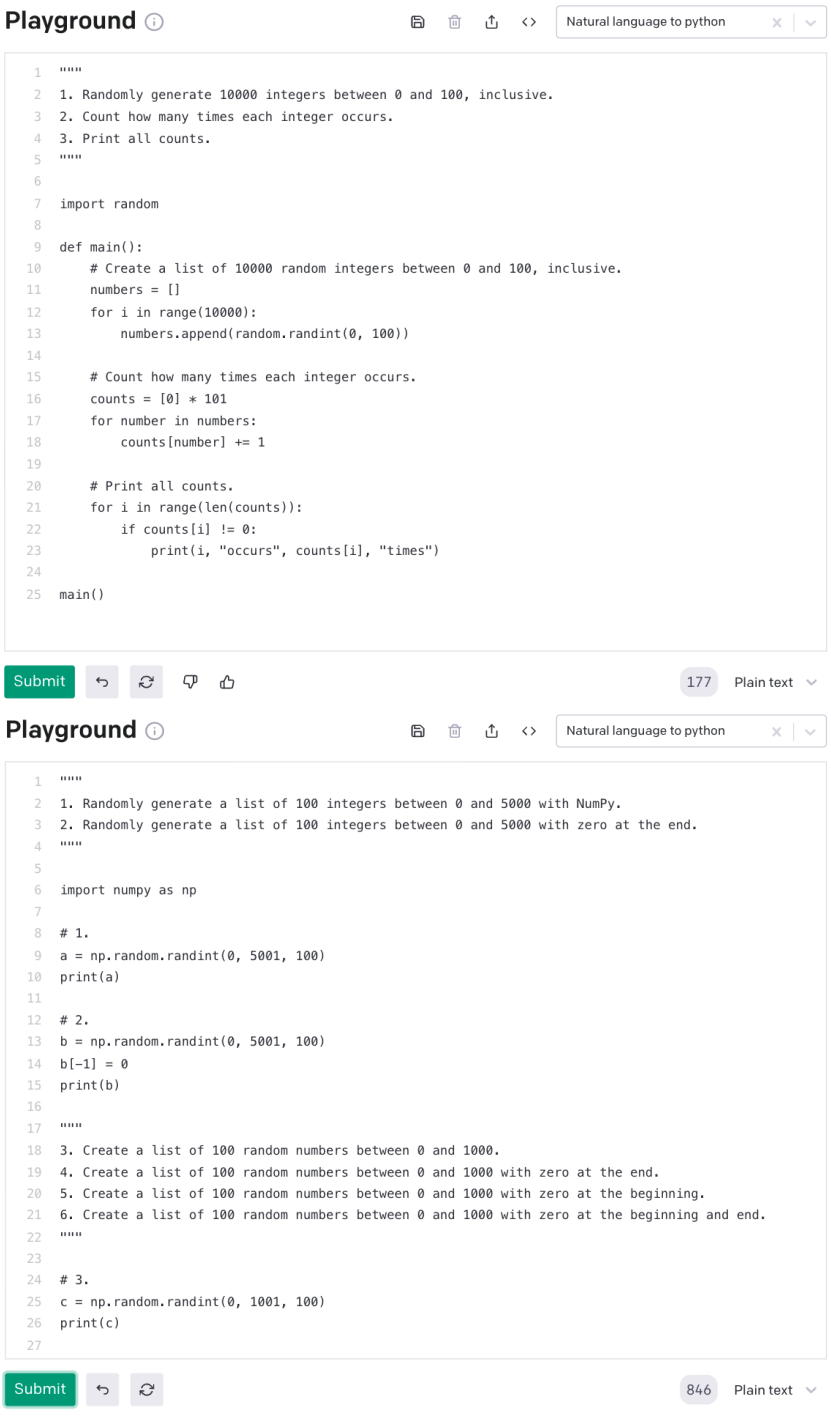


Figure 4·8: OpenAI's Codex is very impressive in many cases (e.g. top), but fails on other cases that the average human programmer would regard as trivial (e.g. bottom).

4.4.2 Adding Context-dependent Commonsense Inference

A notable ongoing work on context-dependent commonsense inference is LM-GAN, my collaborative effort that directly follows from the previous work of RetroGAN (Colon-Hernandez et al., 2021). LM-GAN simply stands for Language Model Generative Adversarial Network. It takes advantage of the same adversarial training as in our RetroGAN work, and uses language model as its Generator component to generate commonsense assertions from texts.

The fundamental limitation of RetroGAN is the lack of contextualization in its generated knowledge embeddings. In other words, had there been different discourses that act as different contexts for a particular word and had RetroGAN been used to generate an embedding for that word, RetroGAN would always generate the same embeddings for the word and not be able to pick up the different contexts.

In LM-GAN, on the contrary, our goal is that the system will generate context-informed commonsense assertions, initially in the form of vectors but immediately converted to symbols for human readability. More specifically, LM-GAN first utilizes a generative language model to conduct commonsense inference over GLUCOSE-style (Mostafazadeh et al., 2020) story-sentence pairs, where the story serves as the context for the sentence. The language model learns to take hints on which components of a commonsense assertion it should generate, and whether to generate more general or more specific versions of an assertion. And due to its context-dependent, the language model can generate different commonsense assertions for the same sentence that is embedded in different story contexts. After that, the GAN is utilized as a variant of the *post-specialization* technique in AuxGAN (Ponti et al., 2018) and RetroGAN (Colon-Hernandez et al., 2021) to help the commonsensically fine-tuned language model to generalize beyond what it has learned. Figure 4·9 shows the complete architecture of LM-GAN:

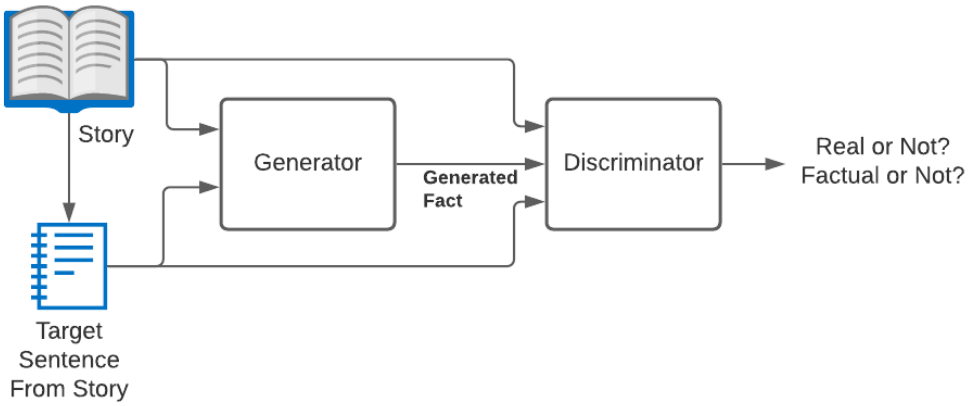


Figure 4·9: Overview of current version of LM-GAN.

Note especially that the Discriminator is geared with both the skill to tell whether a generated assertion is real and the skill to tell whether a generated assertion is factual. Here, a real assertion means one that can be found in any of the original commonsense knowledge bases; a factual assertion may not be found in any existing knowledge base, but nonetheless it may speak some truth. To tell whether an assertion is real, we have simply used the adversarial loss inspired by that of (Goodfellow et al., 2014). To tell whether an assertion is factual, we implemented loss function terms inspired by the work of (Li et al., 2016) that makes use of *negative examples* to prime the GAN to tell apart what is factual from what is counterfactual.

Working Example

To better understand the intricate mechanisms of LM-GAN, I will briefly walk through an example. Suppose we have the following story:

S1: John is a regular person who has a dog.

S2: John, every day, goes out to walk his dog.

S3: A dog is a man's best friend.

S4: Dogs like to bark at other dogs.

Following the GLUCOSE style of knowledge generation, LM-GAN can generate *specific* knowledge and *general* knowledge from this story, by focusing on a specific sentence in the story, and by utilizing *hints*.

Suppose LM-GAN is set to focus on S2 and is given the hint `<CapableOf>`, a ConceptNet relation. Then, LM-GAN can generate the following specific and general commonsense assertions:

specific: John is capable of walking his dog.

specific: John has a dog.

specific: John has a dog enables John to walk his dog.

general: PersonX who has ThingA enables PersonX to walk ThingA.

Figure 4·10: Some examples of specific and general assertions that LM-GAN is capable of generating, provided with a story, a targeted sentence, and a hint. Note that for the general assertion, I used Atomic-style variable names here for clarity.

Note that generating the general knowledge here requires pulling together all the specific knowledge, as well as knowing that John is a PersonX and dog is a ThingA. Note also that linguistic templates are used to generate knowledge in this kind of human-readable format.

Preliminary Analysis

Figure 4·11 shows some preliminary results of our current exploration of LM-GAN.

Model	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L-SUM	BLEU	METEOR
+ ADV + CONF	43.656	10.544	40.380	40.379	31.335	61.683
+ ADV – CONF	**43.747**	10.559	**40.530**	**40.531**	31.279	61.623
– ADV + CONF	43.715	**10.680**	40.292	40.292	**31.470**	**61.776**

Figure 4·11: Preliminary adversarial results of LM-GAN.

These results show that LM-GAN strikes a balance between precision and recall and can be scaled up to do well in both. Of course, in commonsense inference, both precision and recall are vital: invoking lots of wrong knowledge alongside lots of right knowledge (low precision) is troublesome; invoking mostly right knowledge but not enough of it (low recall) is just as troublesome.

Chapter 5

General Related Work

5.1 The State of Natural Language Processing

It is widely recognized that NLP started with the work of SHRDLU (Winograd, 1972), where a computer is programmed to understand human instructions and then to accordingly carry out actions in its simulated environment. Fast forward to the present, we may construe SHRDLU as a miniature demonstration of what the ultimate NLP should look like — that is, machines communicating with us humans in our languages, and do us favors when we ask them to. The reality is, in the 50 years since SHRDLU, NLP has branched off into several, more concentrated directions, each with its own set of approaches and accomplishments.

Some NLP approaches program computers to explicitly analyze words, sentences, and discourses. On the word level, large-scale semantic-graph databases, such as WordNet (Miller, 1995; Fellbaum, 1998) and ConceptNet (Speer et al., 2017), have been crowd-sourced to capture lexical and commonsense meanings of words and phrases. Using such databases of *knowledge* (a.k.a. knowledge bases), tasks such as Word Sense Disambiguation can be tackled, using statistical models that leverage statistical information within and among the various knowledge bases. For instance, see the work of (Havasi et al., 2010) on Word Sense Disambiguation using ConceptNet. In addition to these knowledge-flavored databases, statistical methods have been directly applied to large-scale corpora to learn so-called *word embeddings*. Word embeddings come in many shapes and forms: if a task demands more global context,

GloVe (Pennington et al., 2014) can be used; if a task demands more local context, word2vec (Mikolov et al., 2013) can be used; or if a task demands sub-word information (e.g. RetroGAN (Colon-Hernandez et al., 2021)), FastText (Bojanowski et al., 2017) can be used. Generally speaking, word embeddings have been shown useful in such important tasks as sentiment analysis and document retrieval.

On the sentence level, syntactic parsers, such as CoreNLP (Manning et al., 2014) and spaCy (Honnibal et al., 2020), have been developed to convert individual sentences into so-called *syntactic parse trees*; a syntactic parse tree of a sentence is a specific organization of all the words in the sentence, according to the syntactic relations between the words themselves and phrases consisted of the words. In addition, semantic parsers, such as SEMPRE (Berant et al., 2013; Berant and Liang, 2014), have also been developed to convert sentences into logical expressions that capture the gist of sentences; this is especially useful when dealing with sentences that are supposed to unambiguously describe entities, entity relations, properties, actions, etc. in and of the physical world.

On the discourse level, a discourse can be either a story or a dialog; therefore, story understanding and conversational systems have been developed to leverage word- and sentence-level information to yield an integral understanding of any given discourse as a whole, rather than as a collection of individual sentences. After all, the goal of NLP is to build computer programs that understand discourses; word- and sentence-level understandings are of service to this ultimate goal.

The more trendy NLP approaches, however, have not seen the same explicit distinctions among words, sentences, and discourses. Instead, they segment an entire corpus into identical-sized sets of tokens and treat the segments as contextual information, where the meanings of the individual words are thought to be embedded within their contexts. Afterwards, so-called Language Models (LM) are built and trained to

encode such words-within-contexts. The end results are a lexicon, where each word has a corresponding vector-based representation, called *word vector* or *word embedding*, that is useful for downstream tasks; and a pre-trained language model that can readily convert seen (and unseen) words and documents into vectors, for downstream tasks. Since the introduction of the Transformer architecture (Vaswani et al., 2017) and all the Transformer-enabled language models such as BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019) and their variants as well as the family of GPTs (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020), old-school word embeddings like word2vec and GloVe are being replaced by these language models. At the time of this writing, these language models serve as backbones for many research endeavors and industrial products — for starters, practically all machine translation software (e.g. Google Translate) and virtual (voice) assistants (e.g. Amazon's Alexa) — and are continued to be scaled up to see where their limits are.

Large language models have proven to work better and are much more convenient to use, compared to their old-school counterparts, and their ongoing developments are almost entertaining to follow on social media these days. Still, building real-world versions of SHRDLU-like systems seems like such a lofty goal that people have not dared to touch in 50 odd years. Indeed, natural language is highly ambiguous and the physical world is vastly more complicated than any computer-simulated world. Much unlike us humans, machines still fundamentally lack any general ways of resolving ambiguities in natural language and in the physical world. Whenever set upon tasks that involve ambiguities, state-of-the-art systems largely ignore the intermediate step of explicitly resolving ambiguities and try to solve the tasks in an end-to-end fashion, hoping for the best. Or, they exploit task-specific statistical patterns that are buried to people and they make educated guesses, hoping for the best. Either way, disambiguation has largely been avoided.

5.2 More Technical Details

In NLP, an AI system is given some input texts along with some downstream natural language tasks, and then the system is asked to perform the said tasks by drawing upon information that it extracts from the input texts. At the time of this writing, the most popular and widely accepted approach to NLP is to directly or indirectly use statistical and/or Machine Learning (ML) — including but not limited to Deep Learning — models. These models are pre-trained on input texts to become a language model, which is then fine-tuned on downstream tasks to boost performance.

Chronologically, statistical language models precede neural ones. Among the most well-known statistical language models are n-gram with TF-IDF and (continuous) bag-of-words (BoW/CBoW). At the moment, however, neural language models really are the default go-to choices for many if not most applications and research.

Very roughly speaking, all state-of-the-art neural language models implement the *Sequence-to-Sequence* (Seq2seq) framework, where the input texts are treated as one contiguous input sequence, and the sequence is first *encoded* by the model, and then *decoded* into an output sequence. The forms and contents of a language model's input and output sequences are basically dependent on the model's designated tasks. As an example, in the task of neural machine translation (NMT), an model's input sequence could be a collection of English documents that have been sequentialized into individual English tokens, with or without punctuation and special tokens; if the target language is, say, Chinese, the model's output sequence would be a sequence of Chinese tokens, with or without punctuation and special tokens. In addition, there must be criteria about how good the translated results are; one universally used criterion is BLEU (Papineni et al., 2002), which computes the percentage of correctly translated words in the output sequence. Although BLEU takes into account neither word order nor linguistic features, it remains the most widely used metrics for neural machine

translation. Nevertheless, there have been some metrics that attempt to emphasize word order, e.g. (Han et al., 2012), and metrics that attempt to emphasize syntactic and semantic features, e.g. (Liu and Gildea, 2005) and (Stanojević and Sima'an, 2014), respectively. As another example of Seq2seq, in the task of dependency-style syntactic parsing (e.g. see (Li et al., 2018)), a language model's input sequence could again be a sequence of English tokens taken from a sentence, and its output sequence would be a nested clause that represents a dependency parse tree. The most commonly used criterion is attachment score, which measures the percentage of words that have been assigned their correspondingly correct heads; the style where dependency labels are also taken into account is called labeled attachment score (LAS), and the style without considering dependency labels is called unlabeled attachment score (UAS) (Nivre and Fang, 2017).

The term "Seq2seq" was first coined in (Sutskever et al., 2014); the paper is frequently cited as the first-ever attempt to use Long Short-Term Memory (LSTM) units (Hochreiter and Schmidhuber, 1997), an improved version of the original Recurrent Neural Networks (RNN) units (Rumelhart et al., 1986), to implement both the *encoder sequence* and *decoder sequence* of a Seq2seq model. LSTM is an improved version of the original RNN, because LSTM is specifically designed to target at the problems of *gradient vanishing* and *gradient explosion*; these problems are inevitable when the module sequence becomes too long, and module sequences do become very long in task settings such as neural machine translation. The introduction of LSTM-based Seq2seq models proved to be unprecedentedly successful at solving NLP tasks.

At the same time, RNNs, even LSTMs, have historically been known to be computationally very costly. To combat high computational costs in training LSTM-based Seq2seq models, (Chung et al., 2014) proposed a new RNN architecture called Gated Recurrent Unit (GRU) (Chung et al., 2014), which removed the memory cell from

a LSTM unit and, instead, exposes the entire hidden state. In practice, GRUs have become more prevalent than LSTM, because GRUs are easier to implement, debug, and train, all while ensuring performance.

Soon after the introduction of GRU, (Bahdanau et al., 2015) proposed the idea of imposing Attention Mechanisms on top of Seq2seq models. Their argument was that not all words within a sequence are of equal importance; therefore, there should be a learning algorithm that learns, at each time step, to associate a corresponding weight (a.k.a. attention score) for the current word, by observing the current word in the context of the hidden states of the previous encoder modules.

Because RNN remained computationally expensive and, simultaneously, attention mechanism began giving rise to superior performance in NLP applications across the board, NLP researchers started exploring ways to move closer to attention mechanism and further from conventional RNNs. This shift eventually culminated in (Vaswani et al., 2017), which proposed the idea — as they stated verbatim in their title — that "attention is all you need." With only attention mechanism and no RNN, the new architecture of *Transformer* was born. Today, most large language models in application are derived from the Transformer architecture. Perhaps the two most prominent families of such language models are the Generative Pre-trained Transformer (GPT) family (Radford et al., 2018; Radford et al., 2019; Brown et al., 2020) maintained by OpenAI, and the Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2019) family pioneered by Google and advanced by many. GPT is different from BERT in that information flows from left to right in GPT, but bidirectionally in BERT (see Figure 2.1). As a result, GPT is a *generative* language model that generates a sequence of tokens of some specified length, L, by sampling from the internal state, h, of the model and conditioning on all previously generated tokens at

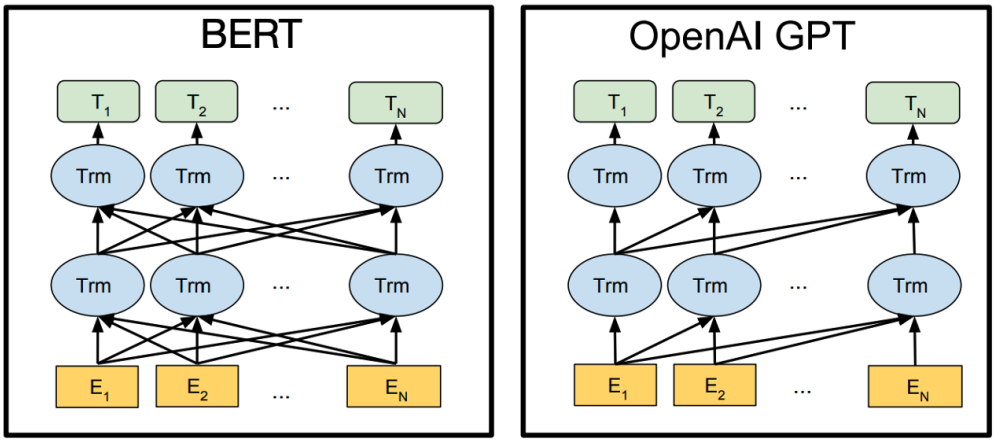


Figure 5·1: Information flows are from left to right in GPT (right), but bidirectionally in BERT (left).

every time step, as in equation (1.1),

$$Pr\{x^{(1)},\cdots,x^{(L)}|x^{(0)},h\} \propto \prod_{i=1}^{L} Pr\{x^{(i)}|x^{(i-1)},\cdots,x^{(0)},h\} \tag{5.1}$$

where $x^{(0)}$ is typically some *start* token. On the other hand, BERT is primarily a *masked* language model that is trained to use information from both directions to fill in the masked out words in sentences.

5.2.1 Natural Language Understanding

A subarea of NLP is natural language understanding (NLU), where we care not only about the outputs of NLP systems, but also how they produce the outputs — for example, whether they produce their outputs in a transparent and interpretable way.

The terms "NLP" and "NLU" are often used interchangeably in various literature. But for me, there is a key distinguishing factor: NLP systems are more task-oriented, whereas NLU systems are more transparency- and interpretability-oriented. On one hand, if an NLP system performs at a superhuman level but fails to make clear how it reaches its results, then determining whether there is *real understanding* in its task-

solving pipeline is difficult. Assuming there is indeed real understanding somewhere, locating precisely where and making sense of its constituents are difficult, too. On the other hand, even if a system does not yield impressive results but, for the ones it does yield (including the wrong ones), a human can see clearly its reasoning processes, then it surely deserves to be called an NLU system.

Previously on NLP, I briefly mentioned Syntactic Parsing, saying that it can be and has been tackled with the general Seq2seq approach using any state-of-the-art language model of choice. Although it technically belongs to NLP, I will expand on Syntactic Parsing in this section on NLU. My rationale is, once again, rooted in the notion of tree-structured Inner Language representations of natural languages (NL) in the human brain (Berwick and Chomsky, 2015). In Syntactic Parsing, an NL sentence is converted by a *syntactic parser* into a tree-structured representation, called *syntactic parse tree*. It remains an open research question as to how much "semantics" can be revealed by syntactic parse trees — or, said in another way, how can syntactic parse trees be made to reveal more semantics — in NL sentences (Mueller, 2002). Nevertheless, these trees are a good starting point for building "Inner Languages" in AI systems.

The two most prominent styles of Syntactic Parsing are Constituency Parsing and Dependency Parsing. In Constituency Parsing, all the tokens in a sentence are treated as leaf nodes, and they are recursively built into higher- and higher-level nodes that represent various phrases, until everything culminates in one node, at the top-most level, that represents the entire sentence. In Dependency Parsing, a sentence is treated as a directed graph whose nodes represent the tokens in the sentence and edges represent, whenever applicable, the *dependency relations* between pairs of tokens. Figure 2.2 shows an example, produced by the Stanford CoreNLP parser (Manning et al., 2014), that contrasts these two styles:

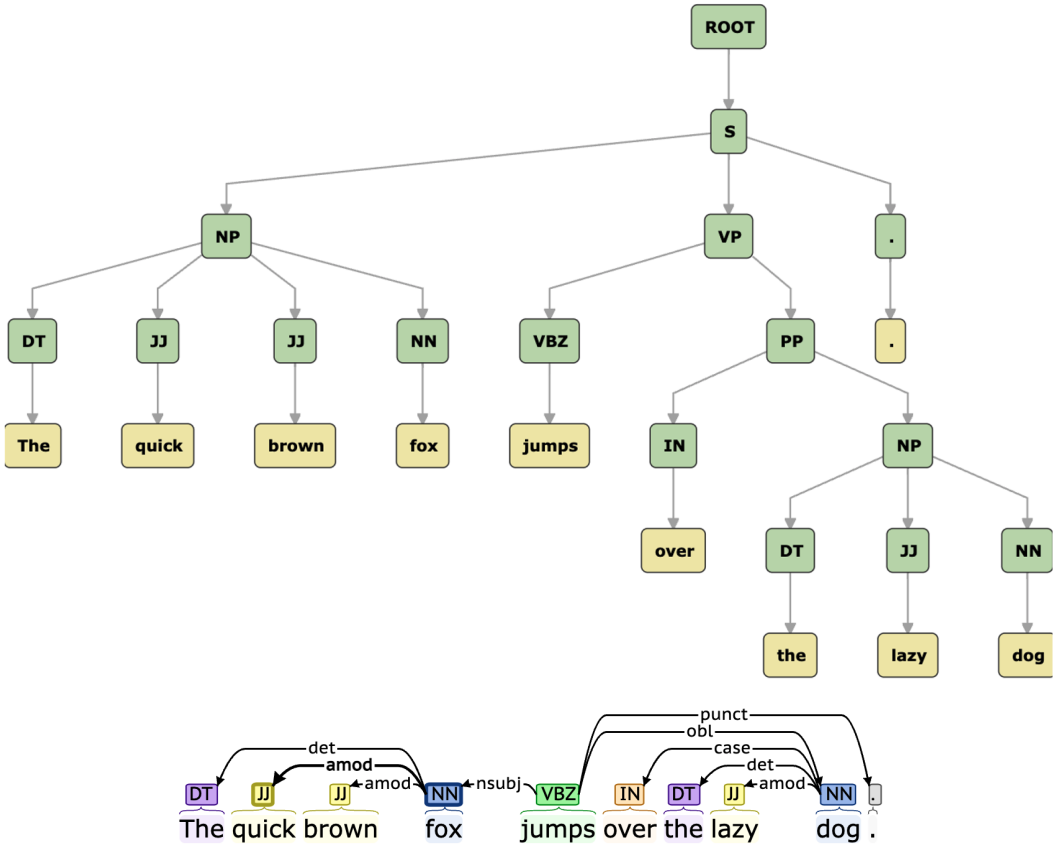


Figure 5·2: Constituency (top) and dependency (below) syntactic parse trees for the sentence "The quick brown fox jumps over the lazy dog," produced by the Stanford CoreNLP parser.

Whether constituency- or dependency-style, syntactic parsers are troubled by two prominent syntactic ambiguities: Prepositional Phrase (PP) Attachment, and Pronoun Coreference. How to resolve these ambiguities robustly and in general-purpose ways, remains an open research question, and there are two primary reasons for that.

5.2.2 Syntax and Semantics

The immediate reason is that, traditionally, Computational Linguistics and NLP researchers isolate the notion of *semantics* (i.e. meaning) from the notion of syntax, and treat semantics as either a downstream task or future work. This isolation is a problem for both PP Attachment and Coreference — both are what they categorize

as syntactic parsing tasks — because the resolutions of both require semantic information. And very likely, the required semantic information is *commonsense*, which is generally not explicitly provided in the text. Consider the following example sentence for PP Attachment:

`The doctor treated the veteran with disabilities.`

This sentence minimally and strictly abides by the (S, V, N_1, P, N_2) syntactic constraint characterized in (Ratnaparkhi et al., 1994), where S is subject, V is verb, N_i are nouns, and P is preposition. Together, P and N_2 comprise the PP, which here is "with disabilities." The task is to determine whether the PP should be attached to V (or, equivalently, to S, assuming that V is S's action) or to N_1. Evidently, using syntactic information alone, it is impossible for a syntactic parser to make an informed attachment decision. Knowing as much (or *little*, depending on one's perspective) as that veterans are more likely than doctors to be associated with disabilities, however, a syntactic parser can justifiably make the more "correct" attachment decision that "with disabilities" attaches to "veteran."

Similarly, consider the following example for Pronoun Coreference:

`The fish ate the worm because it was hungry.`

Abiding by the $(\cdots \mathit{Entity}_1 \cdots \mathit{Entity}_2 \cdots \mathit{pronoun} \cdots)$ syntactic constraint and knowing that eating is related to hunger, a syntactic parser can "correctly" decide that *it* refers to the fish.

In general, by putting *semantics* back into syntax, parsers can begin to understand language in ways that also make sense to human users.

5.2.3 Natural Language Is Intrinsically Ambiguous

The deeper reason that resolving such linguistic ambiguities remains an open research question, is that natural language is intrinsically ambiguous. In the case of preposi-

tional phrase attachment, consider the following, classic example:

```
Jesse saw the person on the hill with a telescope.
```

Without further context, it is unclear who is on the hill and who has the telescope, and therefore, it is intrinsically ambiguous where the prepositional phrases "on the hill" and "with a telescope" should attach to.

Similarly, consider again the same birthday party story as in chapter 4:

```
Robbie and Susie are going to Marvin's birthday party.
One of them wanted to buy a kite.
"But he has one," he said, "he will make you take it back."
```

There are many commonsense knowledge and reasoning pieces that underlie a preschooler's ability to almost instantaneously know which entity each pronoun in the story refers to. Just to name a few salient ones:

- Robbie is probably a boy, Susie probably a girl, and Marvin probably a boy.
- The kite is a birthday gift to Marvin, because it is customary that the host of a birthday party be bought birthday gifts.
- "Them" refers to Robbie and Susie but not Marvin, because it makes sense for invitees of a birthday party to buy birthday gifts for the host, but it does not make sense for the host himself to buy himself a birthday gift.
- It is not immediately clear whether the first "one" refers to Robbie or Susie, until further context is provided.
- Quotes indicate a conversation takes place. The conversation is between Robbie and Susie, because they want to buy a gift for Marvin and it makes sense that

they are communicating about what gift to buy. Therefore, the "he" in "he said" must be Robbie.

- Because Robbie would not refer to himself as "he" in a direct quote from himself, and because the only other male character of relevance is Marvin, the "he" in Robbie's quotes must be Marvin.

- The second "one" refers to "a kite," because the "he" in "he has one" refers to Marvin, "has" indicates ownership, and so far kite is the only object of relevance to ownership.

- "But" indicates that Robbie disagrees with Susie on buying a kite for Marvin because "he has one." Therefore, it must be Susie who wanted to buy a kite, so now we know the first "one" refers to Susie.

- Finally, in "he will make you take it back," "he" is once again Marvin because Robbie is just continuing his sentence; "you" is Susie, because the conversation is between Robbie and Susie, so when Robbie uses "you," he is addressing Susie; and "it" refers to the hypothetical kite that Susie wanted to buy but Robbie disagreed with.

A notable, ongoing work that has addressed such syntactic ambiguities with semantic information, is the START system (Katz, 1997). START has a constituency syntactic parser as its basis and parses sentences into a list of triples that include all the syntactic and semantic information that START can extract from those sentences.

For example, for the sentence

```
The bird flew to the tree because a cat appeared.
```

START produces the following list of triples[1]:

[1]Truthfully, in START, entities and verbs are indexed; for clarity, here we omit indices.

```
[fly because appear]
[bird fly null]
[fly to tree]
[cat appear null]
[fly has_tense past]
[fly has_position leading]
[because is_clausal yes]
[because is_main yes]
[appear has_tense past]
[appear has_clause_type tensed]
[bird has_det definite]
[to has_position trailing]
[tree has_det definite]
[cat has_det indefinite]
[fly has_person 3]
[appear has_person 3]
[bird has_number singular]
[null has_category nil]
[tree has_number singular]
[cat has_number singular]
```

START's parsing capability can be used by downstream NLU systems to understand stories. The Genesis Story Understanding System (Winston and Holmes, 2018) is one such system that takes advantage of START's parsing results and analyzes simple English stories that are carefully written, to reflect and model the human story understanding processes, and to further model human intelligence.

In Genesis, the first step is to convert START triples into Genesis's own Inner Language representations, which is affectionately named *Innerese*. Innerese is directly inspired by Case Frames (Fillmore, 1968), a frame-semantic (Minsky, 1974) representation for language. Using the previous sentence as an example, Genesis produces the following Innerese expression from the previous list of triples:

In addition to representing such input texts in Innerese, Genesis also expresses its

```
(relation cause
          (relation appear
                    (entity cat))
          (relation fly
                    (entity bird)
                    (function to (entity tree))))
```

Figure 5·3: Genesis's Innerese representation for the sentence "The bird flew to the tree because a cat appeared," which has been slightly simplified for readability.

story-processing mechanisms entirely in Innerese. The story-processing mechanisms include various **If-Then** commonsense rules that capture commonsense knowledge, as well as various Script-like (Schank and Abelson, 1977) concept patterns that capture story-level concepts. An example commonsense rule is: If an entity `A` is dead, then among other things, Genesis knows that `A` cannot be happy. An example concept pattern is: If, in a story, an entity `A`'s harming an entity `B` eventually leads to `B` harming `A` back, then Genesis concludes that the concept "Revenge" has taken place in the story.

At an even higher level, because Genesis represents both its data and its story-processing mechanisms in Innerese, Genesis has the ability to tell a story about how it found a concept in a given story, look at the story it tells itself, and then do a higher-level analysis on its self-told story — and Genesis can do so recursively, without any apparent theoretical limit.

NLU systems like Genesis deserve to be merited as Human-Centered AI (HCAI) systems, because their priorities are precisely *transparency*, *explainability*, and *interpretability*. Their working assumption inspires the main argument of this thesis, which once again is: Language will be a key conveyance of human-AI communications, therefore we had better make sure that AI systems understand (and speak) our

languages in ways that are transparent, explainable, and interpretable to us.

Despite the value such human-centered NLU systems, they are yet to be deployed in applications of any scale, because their capabilities are limited — at each level at which they dabble. Their capabilities are limited, because, for a system like Genesis, dealing with unconstrained natural language inputs is a major hurdle. For example, Genesis uses START to process natural language inputs, but START, as a parser, suffers from the same prepositional phrase attachment ambiguities and pronoun coreference ambiguities that were discussed earlier. Unless and until there become robust ways to *scale up* START's and/or Genesis's NLP ability to "translate" multilingual natural language inputs into Innerese representations, Genesis is limited to carefully written, simple English stories that are pedagogical and enlightening, but not broadly useful. Perhaps someday, something like my dissertation work here can find its way to Genesis and take it into the real world.

Chapter 6

Contributions

In this dissertation, I argued that the way to transparent AI is through transparent natural language processing (NLP), and that disambiguation is the key to making NLP transparent. Targeted at disambiguation, a system naturally finds its way to utilizing commonsense knowledge and inference mechanisms. I summarize my overall scientific vision in the following *Commonsense Disambiguation Hypothesis*:

> Disambiguation is the application of commonsense inference to language understanding. In addition, disambiguation at all levels of language will solve most, if not all, of the longstanding problem machine language understanding.

To lay the groundwork for testing my hypothesis, I made the following contributions:

- Implemented PatchComm, and showed that it leverages context-independent commonsense to better resolve sentence-level syntactic ambiguities than purely syntactic parsers alone.

- Implemented ProGeneXP, and showed that it brings together pre-trained language context and newly encountered sentences, to yield transparent descriptions for human users while improving performance on downstream tasks over baselines.

- Implemented DialComm, and showed that it builds semantic representations

for discourses, enabling both sentence-level disambiguation and downstream end-user programming in natural language.

- Incorporated commonsense, in various ways, into language understanding, across various scenarios and tasks.

The results from my own work and my collaborative work suggest to me that these work have successfully set the stage for future advances in NLP and commonsense inference. To achieve future advances, this or another similar line of work needs to be continued. Ultimately, the goal is to build AIs that we can talk to and trust. People talk to each other in natural language, so machines that we build should do the same. People can establish trust with each other just by talking, so machines should do the same. If anything, my dissertation has shown how feasible it is to get started on building transparent AI that understands commonsense. As long as AI people do get started on this — and many are — I have no doubt that the world will be a better place and scientists will finally be able to solve many lingering scientific mysteries.

References

Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98/COLING '98, page 86–90, USA. Association for Computational Linguistics.

Beaulieu, S., Frati, L., Miconi, T., Lehman, J., Stanley, K. O., Clune, J., and Cheney, N. (2020). Learning to continually learn. In Giacomo, G. D., Catalá, A., Dilkina, B., Milano, M., Barro, S., Bugarín, A., and Lang, J., editors, *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, volume 325 of *Frontiers in Artificial Intelligence and Applications*, pages 992–1001. IOS Press.

Belinkov, Y., Lei, T., Barzilay, R., and Globerson, A. (2014). Exploring compositional architectures and word vector representations for prepositional phrase attachment. *Transactions of the Association for Computational Linguistics*, 2:561–572.

Berant, J., Chou, A., Frostig, R., and Liang, P. (2013). Semantic parsing on Freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1544, Seattle, Washington, USA. Association for Computational Linguistics.

Berant, J. and Liang, P. (2014). Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425, Baltimore, Maryland. Association for Computational Linguistics.

Berwick, R. C. and Chomsky, N. (2015). *Why Only Us: Language and Evolution*. The MIT Press.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H., editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating large language models trained on code.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Clark, K. and Manning, C. D. (2016a). Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2256–2262, Austin, Texas. Association for Computational Linguistics.

Clark, K. and Manning, C. D. (2016b). Improving coreference resolution by learning entity-level distributed representations. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 643–653, Berlin, Germany. Association for Computational Linguistics.

Clemenson, G. D. (1977). A birthday party frame system. Technical Report MIT Artificial Intelligence Laboratory Working Papers, WP-140, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA. This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

Colon-Hernandez, P., Lieberman, H., and Havasi, C. (2019). Does a dog desire cake?-expanding knowledge base assertions through deep relationship discovery. In *33rd*

Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

Colon-Hernandez, P., Xin, Y., Lieberman, H., Havasi, C., Breazeal, C., and Chin, P. (2021). RetroGAN: A cyclic post-specialization system for improving out-of-knowledge and rare word representations. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2086–2095, Online. Association for Computational Linguistics.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Faruqui, M., Dodge, J., Jauhar, S. K., Dyer, C., Hovy, E., and Smith, N. A. (2015). Retrofitting word vectors to semantic lexicons. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1606–1615, Denver, Colorado. Association for Computational Linguistics.

Fellbaum, C., editor (1998). *WordNet: An Electronic Lexical Database*. Language, Speech, and Communication. MIT Press, Cambridge, MA.

Fillmore, C. J. (1968). The case for case, dins. In Bach, E. and Harms, R., editors, *Universals in Linguistic Theory*. Holt, Rinehart, and Winston.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

Gerz, D., Vulić, I., Hill, F., Reichart, R., and Korhonen, A. (2016). SimVerb-3500: A large-scale evaluation set of verb similarity. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2173–2182, Austin, Texas. Association for Computational Linguistics.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Han, A. L. F., Wong, D. F., and Chao, L. S. (2012). LEPOR: A robust evaluation metric for machine translation with augmented factors. In *Proceedings of COLING*

2012: Posters, pages 441–450, Mumbai, India. The COLING 2012 Organizing Committee.

Havasi, C., Speer, R., and Pustejovsky, J. (2010). Coarse word-sense disambiguation using common sense.

Hill, F., Reichart, R., and Korhonen, A. (2015). Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics*, 41(4):665–695.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Honnibal, M., Montani, I., Van Landeghem, S., and Boyd, A. (2020). spaCy: Industrial-strength Natural Language Processing in Python.

Javed, K. and White, M. (2019). Meta-learning representations for continual learning. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Katz, B. (1997). Annotating the world wide web using natural language. In *Computer-Assisted Information Searching on Internet*, RIAO '97, page 136–155, Paris, FRA.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Lee, K., He, L., Lewis, M., and Zettlemoyer, L. (2017). End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 188–197, Copenhagen, Denmark. Association for Computational Linguistics.

Levesque, H., Davis, E., and Morgenstern, L. (2012). The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, Proceedings of the International Conference on Knowledge Representation and Reasoning, pages 552–561. Institute of Electrical and Electronics Engineers Inc. 13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012 ; Conference date: 10-06-2012 Through 14-06-2012.

Li, X., Taheri, A., Tu, L., and Gimpel, K. (2016). Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Li, Z., Cai, J., He, S., and Zhao, H. (2018). Seq2seq dependency parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3203–3214, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Lieberman, h. and Liu, H. (2006). Feasibility studies for programming in natural language. In *End User Development*.

Liu, D. and Gildea, D. (2005). Syntactic features for evaluation of machine translation. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 25–32, Ann Arbor, Michigan. Association for Computational Linguistics.

Liu, H. and Lieberman, H. (2005). Metafor: Visualizing stories as code. In *Proceedings of the 10th International Conference on Intelligent User Interfaces*, IUI '05, page 305–307, New York, NY, USA. Association for Computing Machinery.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *ArXiv*, abs/1907.11692.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., 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, Baltimore, Maryland. Association for Computational Linguistics.

Mihalcea, R., Liu, H., and Lieberman, H. (2006). Nlp (natural language processing) for nlp (natural language programming). In Gelbukh, A., editor, *Computational Linguistics and Intelligent Text Processing*, pages 319–330, Berlin, Heidelberg. Springer Berlin Heidelberg.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Minsky, M. (1974). A framework for representing knowledge. Technical report, USA.

Mostafazadeh, N., Kalyanpur, A., Moon, L., Buchanan, D., Berkowitz, L., Biran, O., and Chu-Carroll, J. (2020). GLUCOSE: GeneraLized and COntextualized story explanations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4569–4586, Online. Association for Computational Linguistics.

Mueller, E. T. (2002). Story understanding. In Nadel, L., editor, *The Encyclopedia of Cognitive Science*. Macmillan.

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

Nakashole, N. and Mitchell, T. M. (2015). A knowledge-intensive model for prepositional phrase attachment. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 365–375, Beijing, China. Association for Computational Linguistics.

Nivre, J. and Fang, C.-T. (2017). Universal Dependency evaluation. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 86–95, Gothenburg, Sweden. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Pilehvar, M. T., Kartsaklis, D., Prokhorov, V., and Collier, N. (2018). Card-660: Cambridge rare word dataset - a reliable benchmark for infrequent word representation models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1391–1401, Brussels, Belgium. Association for Computational Linguistics.

Ponti, E. M., Vulić, I., Glavaš, G., Mrkšić, N., and Korhonen, A. (2018). Adversarial propagation and zero-shot cross-lingual transfer of word vector specialization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 282–293, Brussels, Belgium. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. (2019). Language models are unsupervised multitask learners.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer.

Rahman, A. and Ng, V. (2012). Resolving complex cases of definite pronouns: The Winograd schema challenge. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics.

Ratnaparkhi, A., Reynar, J., and Roukos, S. (1994). A maximum entropy model for prepositional phrase attachment. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Sakaguchi, K., Le Bras, R., Bhagavatula, C., and Choi, Y. (2020). Winogrande: An adversarial winograd schema challenge at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8732–8740.

Scha, R. J., Bruce, B. C., and Polanyi, L. (1986). Discourse understanding. Technical Report No. 391, Center for the Study of Reading, Champaign, Illinois, USA.

Schank, R. and Abelson, R. (1977). *Scripts, Plans, Goals, and Understanding*. Hillsdale, NJ: Earlbaum Assoc.

Schuler, K. K. and Palmer, M. S. (2005). *Verbnet: A Broad-Coverage, Comprehensive Verb Lexicon*. PhD thesis, USA. AAI3179808.

Shwartz, V., West, P., Le Bras, R., Bhagavatula, C., and Choi, Y. (2020). Unsupervised commonsense question answering with self-talk. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4615–4629, Online. Association for Computational Linguistics.

Speer, R., Chin, J., and Havasi, C. (2017). Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Speer, R., Havasi, C., and Lieberman, H. (2008). Analogyspace: Reducing the dimensionality of common sense knowledge. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 1*, AAAI'08, page 548–553. AAAI Press.

Stanojević, M. and Sima'an, K. (2014). BEER: BEtter evaluation as ranking. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 414–419, Baltimore, Maryland, USA. Association for Computational Linguistics.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Tripathy, S., Kannala, J., and Rahtu, E. (2018). Learning image-to-image translation using paired and unpaired training samples. In *Asian Conference on Computer Vision*, pages 51–66. Springer.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. (2017). Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vulić, I. and Mrkšić, N. (2018). Specialising word vectors for lexical entailment. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1134–1145, New Orleans, Louisiana. Association for Computational Linguistics.

Weston, J., Bengio, S., and Usunier, N. (2011). Wsabie: Scaling up to large vocabulary image annotation. In *Twenty-Second International Joint Conference on Artificial Intelligence*.

Winograd, T. (1972). Understanding natural language. *Cognitive Psychology*, 3(1):1–191.

Winston, P. H. (2018). Self-aware problem solving. CMHI Report Number 2, MIT DSpace. Computational Models of Human Intelligence Community.

Winston, P. H. and Holmes, D. (2018). The genesis enterprise: Taking artificial intelligence to another level via a computational account of human story understanding. CMHI Report Number 1, MIT DSpace. Computational Models of Human Intelligence Community.

Xin, Y., Lieberman, H., and Chin, P. (2021). PATCHCOMM: Using Commonsense Knowledge to Guide Syntactic Parsers. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 712–716.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251.

Yida Xin Curriculum Vitæ

Email: yxin@bu.edu

Education

Ph.D. Candidate Dec 2021 – Present

- BULISP Group, Department of Computer Science,
  Boston University, Boston, MA, USA
- Final Dissertation Defense completed on 27 June 2022
- **Visiting Student** Jun 2018 – Present
  - Genesis Group, CSAIL, MIT
    Cambridge, MA, USA

M.S. Computer Science Sep 2017 – Sep 2020

- BULISP Group, Department of Computer Science,
  Boston University, Boston, MA, USA

B.S. Mathematics (cum laude) Sep 2013 – May 2017

- Tulane University, New Orleans, LA, USA
- Annual recipient of Dean's Honor Scholarship
- Previously enrolled in Pre-Health (Pre-Med) while
  pursuing B.A. Economics and B.S. Neuroscience

High school Aug 2010 – Jun 2013

- Portledge School, Locust Valley, NY, USA
- 1^{st} place award in Nassau County Math Tournament

Yida Xin Curriculum Vitæ

Email: yxin@bu.edu

Work/Internship

Leela AI Inc. Jun 2020 – Present
Research Consultant (previously Summer Intern)

- Prototyped Python-based research API
- Prototyped Python-based natural language modules
- Prototyped Python-based GUI
- Continuing to endow Leela core agent with human-like learning, reasoning, and planning capabilities

Techinical Skills

- Natural Language Processing, Machine Learning, Artificial Intelligence
- Python 3.x (PyTorch, TensorFlow 2.x), Java, C, Scheme, ATS, MATLAB
- LaTeX, HTML

Language Skills

- Mandarin (First Native)
- English (Second Native)

Research Agenda

My graduate studies have focused on developing Artificial Intelligence (AI) models that use commonsense knowledge to understand natural languages, and applying this capability to the real world, e.g. industrial and service robotics, and virtual assistants. I believe building commonsense-powered natural language understanding models in lab settings is fundamental and indispensable to manufacturing transparent, interpretable, and human-centered AI products in the real world.

Yida Xin Curriculum Vitæ

Email: yxin@bu.edu

Research Activities

Publications

- **Xin, Y.**, Lieberman, H., and Chin, P. (2021). PatchComm: Using commonsense knowledge to guide syntactic parsers. In *Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning*, pages 712–716.

- Colon-Hernandez, P., **Xin, Y.**, Lieberman, H., Havasi, C., Breazeal, C., and Chin, P. (2021). RetroGAN: A cyclic post-specialization system for improving out-of-knowledge and rare word representations. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2086–2095, Online. Association for Computational Linguistics.

Other Papers

- **Xin, Y.**, Lieberman, H., and Chin, P. (2020). Revisiting the Prepositional-Phrase Attachment Problem Using Explicit Commonsense Knowledge. ArXiv, abs/2102.00924. Submitted to The 28th International Conference on Computational Linguistics, 2020 (COLING'2020).

- **Xin, Y.**, Lieberman, H., and Chin, P. (2021). PATCHCOMMPRO: Commonsense Parsing for Natural Language Programming. Submitted to Intelligent User Interfaces, 2022 (IUI 2022).

- Colon-Hernandez, P., Lieberman, H., **Xin, Y.**, Yin, C., Breazeal, C., and Chin, P. (2021). Adversarial Transformer Language Models for Contextual Commonsense Inference. Submitted to Semantic Web Journal, 2022 (SWJ 2022).

- Colon-Hernandez, P., Lieberman, H., **Xin, Y.**, Yin, C., Breazeal, C., and Chin, P. (2021). Can Language Models Take A Hint? Prompting for Controllable Contextualized Commonsense Inference. (Work in progress.)

Presentations

- How might an intelligent agent describe what it sees and find what it is told? International Workshop on Self-Supervised Learning (IWSSL), 2021.

- Taking story-understanding systems to the real world via better parsing. *Story-Enabled Intelligence Workshop*, Advances in Cognitive Systems 8 (ACS-8), 2019.

Yida Xin Curriculum Vitæ Email: yxin@bu.edu

Other Experience

CELOP, Boston University Apr 2019 – May 2019

- Co-instructor, Computer Science Course for KAUST Foundation-Year Program

Ochsner Health System Jan 2016 – May 2016

- Research volunteer, Ochsner BITE Pain Research study at The Center for Nursing Research

Tulane University

- Undergraduate research assistant, Molix Lab Sep 2015 – May 2016
- Undergraduate research assistant, Robinson Lab Jan 2015 – May 2015

Teaching History

- CS131 Combinatoric Structures, with Vahid Azadeh-Ranjbar (Spring, 2022)
- CS630 Graduate Algorithms, with Steven Homer (Fall, 2021)
- CS542 Machine Learning, with Sarah Bargal (Spring, 2021)
- CS101 Introduction to Computer Science, with Perry Donham (Fall, 2020)
- CS101 Introduction to Computer Science, with Perry Donham (Spring, 2020)
- CS391 Foundations of Data Science, with Dora Erdos (Fall, 2019)
- CS132 Geometric Algorithms, with Abbas Attarwala (Spring, 2019)
- CS132 Geometric Algorithms, with Abbas Attarwala (Fall, 2018)
- CS542 Machine Learning, with Peter Chin (Summer, 2018)
- CS132 Geometric Algorithms, with Vahid Azadeh-Ranjbar (Spring, 2018)
- CS101 Introduction to Computer Science, with Perry Donham (Fall, 2017)