Tyler Osborne

Prof. Amittai Aviram

CSCI4911 Readings in Computer Science

11 June 2022

Cognitive States Project Deep Learning Research: Semester in Review

I. Project Overview

The Cognitive States project is an ongoing investigation of belief state implicit in natural language, and specifically, how well a pre-trained deep learning model fine-tuned with various repositories of annotated text can detect belief state. Overall, the goal of this project is to make incremental progress towards more advanced belief state and sentiment detection capabilities within natural language processing (NLP) deep learning models.

II. Definitions of Key Terms

In linguistics, belief state is the degree to which a speaker believes in the truthfulness of their utterance. When a speaker introduces new information, they communicate more than what they explicitly say; among other things, they also convey how sure they are that what they are saying is true. Of course, a sentence is unlikely to say, for example, "John said Mary is coming to dinner with sixty-five percent confidence." Rather, a listener would piece together John's confidence in Mary's attendance at dinner through metadata such as his tone or other information contained in the surrounding text, thus only necessitating the author to say something like, "John said that Mary might come to dinner," without any information lost versus the unlikely example. There is also the issue of the sentence's author, who is not explicitly mentioned in the sentence, and their level of belief regarding the truthfulness of John's utterance. John and the author of the sentence are thus two distinct *sources*, and their respective claims, which for John is that Mary

will come to dinner and for the author that John said Mary is coming to dinner, are referred to as *targets*; the *head* of the target clause, i.e. the top word of the clause's parse tree, is what is annotated with a belief label. In this example, it is 'said' for the author, and 'coming' for John according to the author. Training examples were extracted from various *corpora*, or collections of text, annotated in such a way so as to link source-target pairs to belief values (i.e., a value dictating a source's level of belief; the actual enumerations differ between corpora), creating a source-target-*label* triple, where the label is the aforementioned belief value. Throughout our research, we have used such source-target-label triples as training data to fine-tune a pre-trained deep learning model that will be described later, where the source and target are given and the label is predicted by the model.

III. Explanation of Relevant Linguistics Theory

Belief state theory is undergirded by theoretical semantics research from the past half-century. Two key papers, Kiparsky & Kiparsky (1970) and Grice (1975), appear as major inspirations for later linguistics research; although our specific goal is much more narrow in scope than what these papers discuss, they nevertheless act as guiding forces for our work and are worth describing here.

We begin with Kiparsky & Kiparsky. This paper postulates two syntactic categories for sentence predicates, *factive* and *non-factive*, and furthermore that the syntax of the predicate is predictable through whether or not the subject's claim can be presupposed to be true. Let us start with an example:

Predicate Type	Sentence
Factive	It is odd that it is raining
Factive	I regret that it is raining.
Non-Factive	It is likely that it is raining.
Non-Factive	I suppose that it is raining.

In the factive examples above, it is clear from the form of the sentence that the speaker or author believes that it is raining. In the non-factive cases, we can make no such assumption. This rule holds for presuppositions where a sentence's author claims something about that which is presupposed; in this example, the sentence author is expressing personal sentiments, i.e., oddity and regret, *about* the fact that it is raining, therefore qualifying it as the type of presupposition falling under the factivity rule. A counter-example could be, "John's being ill is impossible to imagine," because even though John's illness is stated as objective truth, it is not presupposed by the author as denoted by the fact that the sentence is only asserting the truth of the claim and nothing further about it.

Predicates in English always involve a verb; an excerpt of a list of factive and non-factive predicates as shown in the paper appears below.

Factive	Non-Factive
Significant	Likely
Odd	Sure
Tragic	Possible
Exciting	True
Relevant	False
…	…

Currently, our research does not encompass the preservation of contextual information, including presuppositions, between sentences, as our corpora do not include annotations of contexts extending beyond a sentence at a time; however, we see deep learning detection of these sentence paradigms as an eventual goal toward which the Cognitive States project will act as a stepping stone.

Grice also discusses how competent speakers make important inferences from what they hear based on identifiable patterns and structures, but here the patterns are more relevant to how one utterance follows the previous utterance, preserving relevance between them. By breaking the norms of conversational relevance, speakers can communicate ideas without explicitly stating them, thus introducing an *implicature*. Let us look at an example.

Example 1: A driver speaks to a local on the side of the road after running out of fuel.

A: I am out of petrol.

B: There is a garage 'round the corner.

If we assume that both A and B are following conversational norms, then we can deduce that B is not infringing on the common expectation that conversational contributions be relevant. This leads us to conclude that B thinks the garage around the corner is open and selling fuel; this is the implicature.

These implicatures appear in discourse when one or more of a set of proposed guidelines for a conversation are not met for one of a number of reasons. This set of guidelines is called the Cooperative Principle, and its maxims are quantity, quality, relation, and manner. Grice presents some illustrative analogies to these maxims:

Maxim of Cooperative Principle	Definition	Example
Quantity	Be informative, but not overly so.	If I ask for four screws, I expect you to hand me four screws. Not three, not five.
Quality	Do not say things you know are false or lack evidence for.	If I am baking a cake and ask for sugar, I expect you to hand me sugar, not salt.
Relation	Be relevant.	If I am mixing the cake batter, I do not expect to be handed a fascinating book.
Manner	Be brief and unambiguous.	I expect a helper to be clear with what they are doing and to do so in a reasonable amount of time.

Grice's maxims, while precise, simply reflect the implicit ground rules governing how we as humans converse with one another. Essentially, if a speaker violates one of these maxims and there is not an obvious justification, then an implicature has been introduced. One more example appears below for emphasis.

Example 2: Of a man known to have broken up all the furniture, one says "He was a little intoxicated."

Here, the speaker is flouting the quality maxim since he is presented with irrefutable evidence of the drunk man's high level of intoxication but nevertheless disregards the rule of not uttering low-quality information, or information known to be false (i.e., that he was not overly drunk). Therefore, a conversational implicature appears, being that the man was indeed overly drunk.

Again, as with Kiparsky & Kiparsky, this type of work is not something that our current data sources and deep learning libraries can support. However, the ability of computers to be aware of implicit semantic features in natural language (i.e., implicature and

presupposition/factivity) will undoubtedly be major pillars of the future of natural language processing, and our work is nudging the field closer to making that a reality.

IV. Pre-Semester Preparatory Work

Alongside working to understand the two papers I just explained, I engaged with additional resources to prepare myself for work on the project. Considering my lack of previous experience in machine learning, deep learning, and linguistics, I needed to spend a large amount of time ingesting the basics as well as certain technical details surrounding these disciplines over the pre-semester winter break. I began with the Coursera Machine Learning online course taught by Andrew Ng. Working through the videos and quizzes of the first half of the course, I learned key terms and concepts such as the difference between supervised and unsupervised learning, what a loss function is and how to weight certain attributes of training examples to optimize gradient descent, just to name a few. This course is a bit dated now (originally created in 2012), so I supplemented it with the Deep Learning Specialization course, also taught by Andrew Ng. The latter course tasked me with writing vectorized Python code to complete the homework's implementation of gradient descent toward the goal of identifying cat pictures. Working through these online courses taught me the vocabulary and workflow of machine learning projects and gave me enough of a high-level overview of their internal mechanisms that I could follow machine learning implementations in Python.

My theoretical work also continued over the course of the semester. I watched recordings of paper presentations in Prof. Owen Rambow's Computational Linguistics Graduate Seminar at Stony Brook to expose myself to more literature concerned with linguistics and deep learning.

V. Overview of Codebase & Associated Python Libraries

The Cognitive States codebase is written in Python and utilizes the following stack: HuggingFace at the top as an API to Transformers for deep learning, which relies on SKlearn, which depends on PyTorch. We also use Datasets, SQLite, and a number of custom-made modules for data pre-processing. The Datasets package has made it easier to organize examples into training and testing sets and post them on the HuggingFace website for public use, while SQLite is being used to write a database implementation that I will describe later.

HuggingFace is the most critical package, providing us with access to pre-trained NLP models that allow the feeding of additional training data for fine-tuning purposes. We chose the BERT (Bidirectional Encoder Representations from Transformers) model since it is good for NLP tasks where the context around a highlighted word (the target in our case) is necessary to predict the answer (i.e., belief value), as is the case here. For a given corpus, we split the data into two groups: training and testing. Our configuration is such that seventy-five percent of the examples are fed to BERT to fine-tune the model, while the remaining twenty-five percent are used as test cases to see how well the model can predict belief values after the fine-tuning is complete. The predicted belief value is then compared to the actual value, and finally, the model outputs aggregated performance metrics.

VI. Language Understanding (LU) Corpus

At the time that I joined the project, a number of deep learning experiments had already been performed on top of an object-oriented Python implementation. This implementation handled the entire preprocessing, model-feeding, and testing process, with one instance of a Cognitive States Data Structure (`CSDS`) object mapping to one source-target training example, and the fields (sentence, source, target, and belief value) being drawn directly from the flat files of the Language Understanding (LU) corpus. This corpus is relatively simple, with all

annotations being "author-only," meaning the source is always the author of the sentence itself rather than another source embedded in the sentence. John in "John said Mary is coming to dinner" would be an example of an embedded, or nested, source, which is not supported by LU. A separate module named `xml2csds` (`XML` format to `CSDS` object) performs the actual work of looping over LU's files to load the training and testing examples into individual `CSDS` objects before another module named `csds2hf` (`CSDS` object to HuggingFace) handles feeding `CSDS` objects into Datasets objects from the Datasets package, before finally being sent to HuggingFace's subroutines to fine-tune BERT.

VII. Factbank Corpus & Experiment Results

LU is not the only corpus we have been working with. A much richer and more complex corpus is Factbank, which diverges from LU in two key ways. First and most importantly, the data is stored relationally; while in flat files, the data are intended to be ported to a SQL-type relational database and come with columns to be used as primary keys and foreign key columns for joins. We are using SQLite for this. Second, Factbank supports embedded sources and preserves the nesting tree of embedded sources for each sentence. For the sentence "John said Mary is coming to dinner," the `sources` table would have two entries, `AUTHOR` to denote the author of the sentence (every sentence has one entry like this), and one additional entry, `John_AUTHOR`, which denotes "John according to the sentence's author." Every training and testing example then consists of this unique source identifier, an associated target identifier, and finally a belief value.

After a first look at the data made clear that additional logic would be required to extract nested-source training examples in the same format as the author-only ones, we broke the task down and ran an experiment using just Factbank's author-only annotations. The composition of

Factbank is such that over ninety percent of the training examples are labeled with one of two belief values[1]:

Belief Value	Description
CT+ (Certainly Positive)	According to the source, it is certainly possible that X.
Uu (Fully Underspecified)	The source does not express a belief value.

Since there was not enough training data for Factbank's other classifiers within the author-only training examples, our experiment did not find results for them. It is also important to note that the CT+ examples were much more numerous than the Uu ones. Below are our results.

Classifier	Metric	Value
CT+	Precision	0.932
CT+	Recall	0.957
CT+	F1 Score	0.945
Uu	Precision	1.0
Uu	Recall	0.158
Uu	F1 Score	0.273

To interpret these data, we must first understand how these metrics work and what they tell us about BERT's performance. For a given classifier, a true positive arises when the predicted label is correct, and a false positive when it is not. Furthermore, a true negative occurs when the predicted label matches the true label but it is not the one we care about, and finally, a false negative appears when the true label is relevant but the predicted one is incorrect. Precision is defined as the number of true positives divided by the total number of true positives and false

[1] https://www.cs.brandeis.edu/~roser/pubs/fb_annotGuidelines.pdf

positives, representing the percentage of the time that BERT correctly labeled training examples within all of its predictions for a given label. On the other hand, recall is calculated as the number of true positives divided by the total number of true positives and false negatives, meaning the percentage of the time where BERT actually caught training examples with a given label in the sample. Finally, the F1 score is the harmonic average of precision and recall, acting as a combinatory metric of the two.

Somewhat unsurprisingly, the performance of the CT+ class was high since this was the majority class (i.e., the majority of training examples had CT+ as the true label), thus giving deep learning more data to work with. The Uu class performed comparatively worse, although the precision score of 1.0 was good; this can be interpreted as, for every training example, if it was predicted to be Uu, it was correct. The low recall score makes sense because it encompasses any class for the prediction but only one for the true class; the relatively high number of total available classes in the model combined with the low number of Uu training examples coalesce to produce this score as the model ended up with many more false negatives than false positives (this being the key difference between the precision and recall formulae).

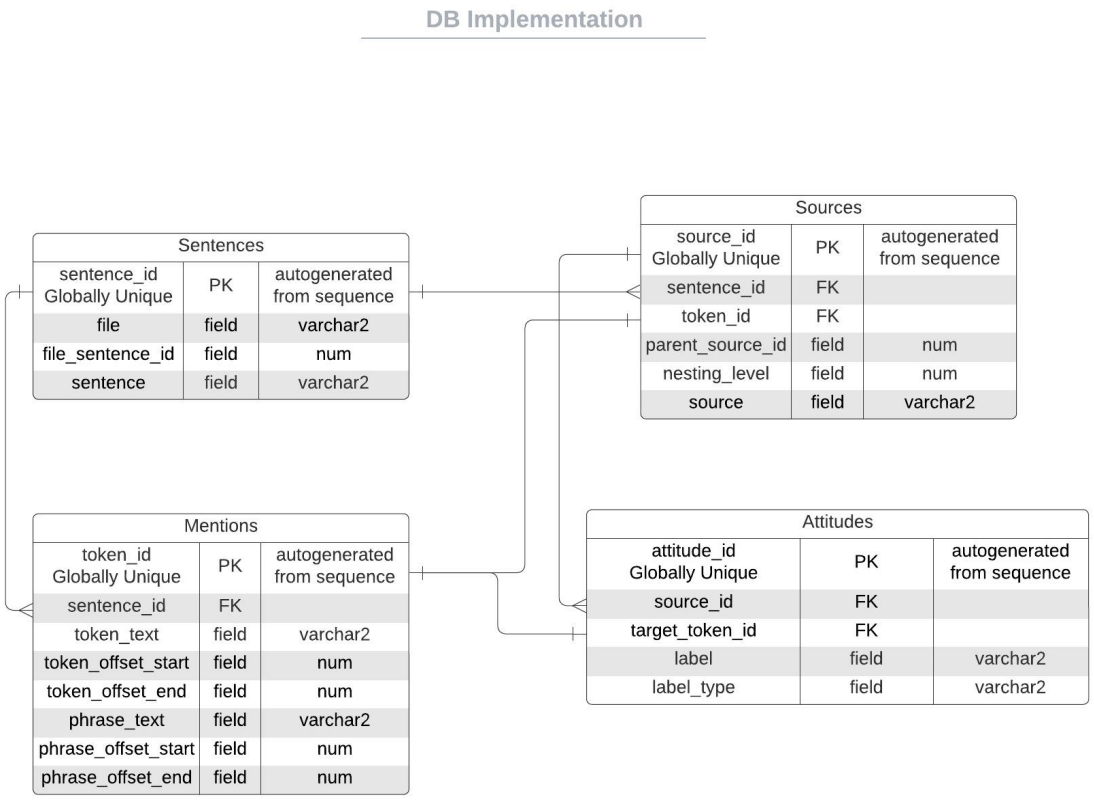
VIII. db2hf 1.0

Implementing the author-only experiment turned out to be relatively straightforward in Python; I named the module `db2hf 1.0` (database to Hugging Face). When I joined the project, a previous student had already ported the corpus' flat files to an SQLite database, allowing us to write one query pulling training examples into a result set for preprocessing before use in BERT. Preprocessing was simple except for the issue of character offsets. BERT needed to know what the string index boundaries were in the sentences of each training example for labeled text (i.e., sources and targets), but these data were stored on a per-file basis rather than a per-sentence basis, thus requiring additional logic to store a dictionary of initial file offsets to subtract off of later ones and then verify that the labeled text matched the text sliced from the sentence using the offsets. I then copied over some previously-written code splitting the preprocessed data into testing and training sets and removing any leftover bugs.

IX. db2hf 2.0

With the Factbank author-only experiment complete, the next step was to bring in the nested-source training examples for additional BERT fine-tuning, marking the second iteration of the module, `db2hf 2.0`. As discussed, we already had an SQLite representation of Factbank, but the structure of the database, as posited by Factbank's authors, turned out to be extremely convoluted and poorly suited for our purposes. This sparked an idea, not only to make using Factbank easier but to streamline the project overall: designing a master SQLite database schema into which we could then port each respective corpus we wish to analyze, only preserving data from original corpora as needed. This would afford a plethora of advantages, one being the standardization of SQL queries across corpora, and another being the prevention of headaches

when others join the project and want to understand our data quickly (otherwise, one would need to learn each corpus' structure in detail). An entity-relation diagram of the schema is below.



Once the master schema was finalized, we endeavored to load in Factbank from the previous database while preserving the validity of each training example, including those with a nested source. My first attempt at this was to tweak and reapply the query we had used for the author-only experiment, removing the query's author-only restriction and filling in the appropriate tables for each row in the result set. However, the nature of the key relationships in Factbank required much more tediously designed and granular logic than what that query provided. Rather than fill in the master schema example by example, we needed to select one sentence and then iterate over the relevant information contained within it, starting with the

author annotation since it is always present (every sentence has an author) and working through each additional level of source nesting, in order. Going back to our first example, "John said Mary is coming to dinner," we would have one annotation for the author at nesting level zero, and then one annotation for John according to the author at nesting level one. Both annotations would have an associated belief value and integer offsets for the source and target; source offsets are non-applicable for the author since the author is implicit and will be zero and four for John since those indices splice 'John' from the sentence string. Target offsets are five and nine for the author since 'said' is the target there, and would surround the word 'coming' for John according to the author. Looking at the schema, this translates to an insert on the `sentences` table first before inserting multiple rows on the `mentions`, `sources`, and `attitudes` tables with the exact number of inserts depending on how many total annotations there are for the given sentence.

X. Current State of the Project, Future Plans & Conclusion

Currently, I am working through SQLite bugs in my current implementation of `db2hf 2.0` that we theorize have to do with executing too many queries inside of nested for-loops. The solution may be to rely more heavily on Python data structures before performing just one round of inserts at the very end to fill in the master database; this is in development now. Once Factbank is loaded into the master database, we will train on that additional data and check the results against the author-only data.

In the long term, we aim to begin work with additional corpora such as MPQA (Multi-Perspective Question Answering), among others. With the master schema in place, it is feasible to imagine a consistent workflow emerging as project work continues: discover a corpus, research its structure, write a one-off Python script porting the files to a fresh copy of the master

schema, and fine-tune BERT on the data using a generalized script. Finally, we would be able to combine corpora together into ever-larger training and testing sets for more effective fine-tuning.