The Text-Minder

Babysitting Your Language Model by Training with Semantic Constraints

Presenter : Jerry Ding (dalud)

Language Models and Their Uses

- A language model learns the probability distribution of word or character sequences in a dataset.
- As a PDF, they can generate data / compute perplexity / build classifiers / perform Bayesian inferences

- Use cases:
  - Autocorrect / text completion
  - Projection to syntactically / stylistically accurate text
    - e.g. for QA, machine translation, conversational agents
  - Generating templates for further human refinement

Shortcomings of Language Models

- Language contains a lot of entropy, i.e. “information”

- **Even with the perfect probability model, and even after conditioning on many things...**

  There are still multiple valid output sentences. These can significantly differ, even in semantic content

- Especially conversation / dialogue. Unlike QA, there isn't just *one true response* to your conversation partner.

GPT-2 Demo

"Oh, hey George! Haven't seen you in a long time. How's it been lately? I heard that you were looking for a new job in San Francisco."

"Haha, yeah we haven't met since graduation. Just hopping on the tech industry bandwagon like everyone else. I self-studied web dev stuff while I was in Virginia."

"Uhh, you were a mechanical engineer the past two years right? How's work? I heard it's tough for entry level positions."

Output #1:
"Yeah, I know what you mean. Work's been pretty hectic as a result of all the new company initiatives, and the new head of the company brought in a few new people who have a bunch of awesome projects under their belts.”

Output #2:
"Well, to tell you the truth, I wasn't sure. I mean, it's just so new here. My coworkers are all like, 'George, didn't you just quit? What the hell are you doing out here?'”

Output #3:
George scratched his head while wiggling his fingers in confusion. "Not exactly. I worked in both IT and Mechanical Engineering for a living for a couple of years. They aren't the most glamorous careers by any means, but...

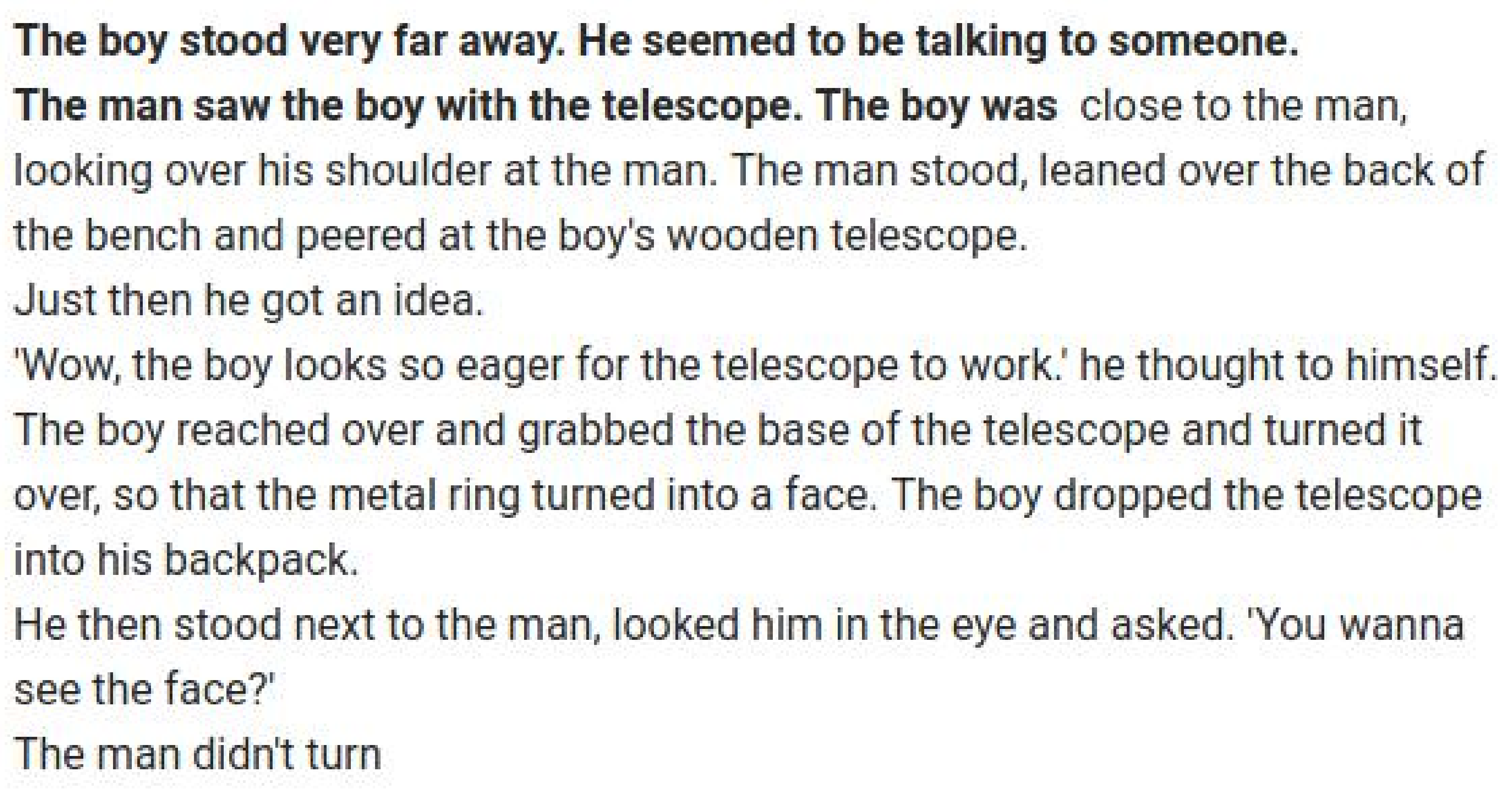
Language Model Conditioning

- Even when conditioning the model on past context...

  It's difficult to inject the right kinds of information to constrain the output of the language model

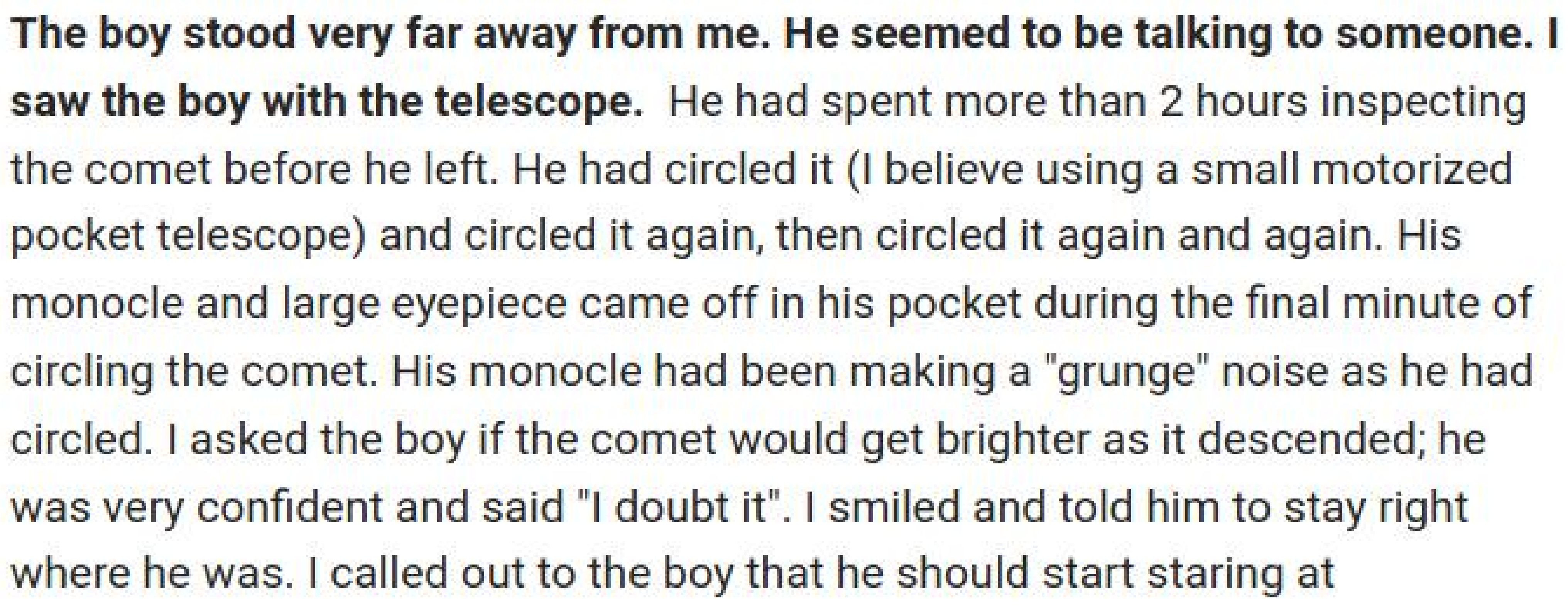
- Can be undesirable even if the text is modeled well
- Much more annoying when the language model makes glaring semantic errors

GPT-2 Conditioning



The boy stood very far away. He seemed to be talking to someone.
The man saw the boy with the telescope. The boy was close to the man, looking over his shoulder at the man. The man stood, leaned over the back of the bench and peered at the boy's wooden telescope.
Just then he got an idea.
'Wow, the boy looks so eager for the telescope to work.' he thought to himself.
The boy reached over and grabbed the base of the telescope and turned it over, so that the metal ring turned into a face. The boy dropped the telescope into his backpack.
He then stood next to the man, looked him in the eye and asked. 'You wanna see the face?'
The man didn't turn

GPT-2 Conditioning



The boy stood very far away from me. He seemed to be talking to someone. I saw the boy with the telescope. He had spent more than 2 hours inspecting the comet before he left. He had circled it (I believe using a small motorized pocket telescope) and circled it again, then circled it again and again. His monocle and large eyepiece came off in his pocket during the final minute of circling the comet. His monocle had been making a "grunge" noise as he had circled. I asked the boy if the comet would get brighter as it descended; he was very confident and said "I doubt it". I smiled and told him to stay right where he was. I called out to the boy that he should start staring at

The Project Question

Is it possible to engineer a method of semantically constraining the output of the language model?

My claim:

One can significantly influence a language model by introducing a small and incomplete encoding of the semantic content desired in the output.

Relevant Prior Work

- The task shares similarities with machine translation
  - Input sentence ≈ semantics desired in the output
  - But in MT, the injected information is a nearly complete characterization of the output

- There is also a similarity to NLGen systems
  - But the motivation and data source is of a different nature
  - NLGen transforms machine data for human interpretability
  - My task: semantic encoding is made by user, and output is intended for communication with other agents

Semantic Constraints

- Semantic encoding: a sequence of **semantic constraints**

- Semantic constraints are simple, non-composable propositions built out of a small vocabulary
  - Basically, a heavily simplified written language

- The constraint is that the above propositions **should be inferable** from the model's output sentence

Semantic Constraints

- **Relevance**: Simply to point out that a noun is mentioned, e.g. `Relevant(door)`
- **Membership**: The noun belongs in a larger class, e.g. `Member(Nicholas, actor)` This can be a hypernym relationship, or information inferred from the textual context.
- **Description**: The noun has a characteristic or is in some state, e.g. `Descr(You, Sad)`
- **Action**: The noun is an agent or experiencer, and the patient is specified if it is also just a noun. The action can be normal or negated, and it can be real or hypothetical, e.g. `Action(I, +help[You, REAL])`. Tokens exist to describe a patient argument that is empty, unknown or complicated (e.g. a thought, event or situation).
- **Recipient**: Reversed relationship of an action with the same slots, e.g. `Patient(Window, +break[UNK, REAL])`

Approach

1. Develop a representation for the semantic constraints
   - Design the grammar of the constraint language, i.e. define the abstract types, data structures, and their arguments
   - Build a small vocabulary for the tokens in the constraint language, making a trade-off between the simplicity of the language and the expressiveness of the constraints

[Much of this is already done]

Approach

2. Create an inference engine to produce semantic constraints from sentences
   - Create a function to map words (from a large vocabulary) to a set of terms in the reduced vocabulary (with similarity scores)
   - Parse each sentence with existing tools, e.g. TUPA
   - Write soft logic rules to infer properties of the words, based on the semantics between terms in the reduced vocabulary
   - Generate semantic constraints from the inferred properties

[Already did the first task]

Approach

3. Train the constrained language model
   - Generate semantic constraints from sentences in corpus
   - Obtain pretrained GPT-2 model, and add encoder / decoder layers around the trained portion
   - Train the model, using the machine-generated semantic constraints as training data for the conditioning process
   - Measure the text perplexity of the trained model vs. an unconditioned or less conditioned model

The Data

- Using the OpenSubtitles dataset
  - Movie lines approximate conversation
  - Full dataset has 280M lines, took a subset of 3M lines
- Will be using pre-trained GPT-2 medium size model
- Used WordNet as an ontology for mapping words to the reduced vocabulary

Sources of Semantic Information

- Manually created
  - Design of the constraint language
  - Creating the reduced vocabulary
  - Inference rules with the reduced vocabulary
  - Tuning parameters for the word mapper

- Existing sources
  - WordNet for synonymy and ontology
  - Pretrained GPT-2 weights
  - GloVe vectors (only as a reference in making reduced vocab)

Appendix

Thoughts on the work done so far

Work so far: Thoughts

- Fun exercise: manually extracting semantic constraints for example sentences
  - To test the expressiveness of the constraint language
  - Observation: even without composition, a simple language can encode a surprising amount of information

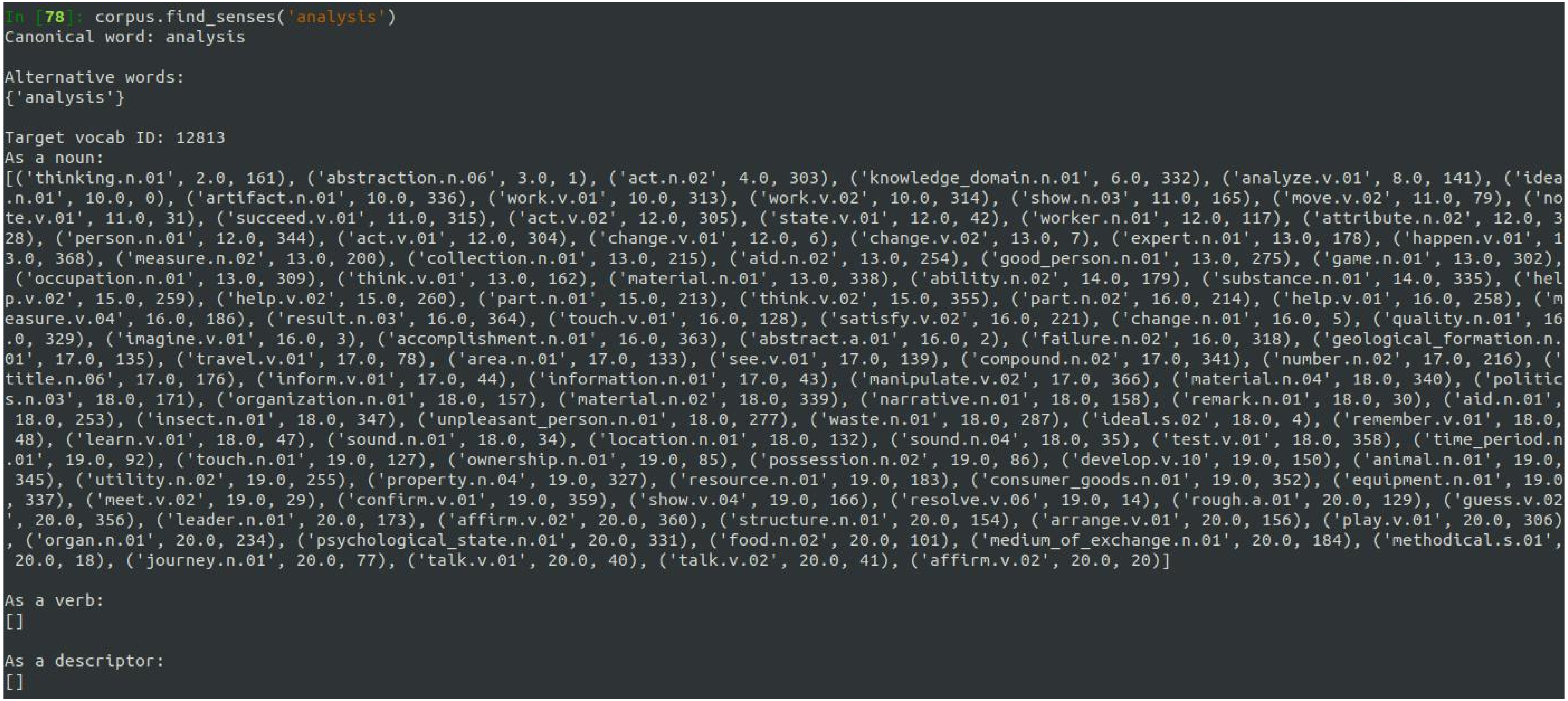
Work so far: Thoughts

- Not easy: condensing down the vocabulary
  - Words don't nearly fit into an ontology
  - Common words are not very informative
    e.g. it's useless to group all concepts under “entity” / “object” / “abstraction”
  - K-means on word vectors do reveal interesting patterns.
    But the clusters aren't useful as primitive terms.
    Clusters often have mixed P.O.S. and antonyms

Work so far: Thoughts

- Insightful: implementing the word mapper
  - This teaches a lot about the structure of WordNet
  - Discovering firsthand why word similarity is a difficult problem
    - Relationships across P.O.S are very sparse
    - Graph connectivity is low with just hypernym / hyponym relations, resulting in underestimated similarity between words
    - Can't tell whether two word senses are very similar, or totally different
    - No concept of prototypical meaning; words are often related via obscure synonyms or word senses
    - Lack of consistency: cannot rely on relationships existing between similar concepts even with the same P.O.S

Word Mapper



```
In [78]: corpus.find_senses('analysis')
Canonical word: analysis

Alternative words:
{'analysis'}

Target vocab ID: 12813
As a noun:
[('thinking.n.01', 2.0, 161), ('abstraction.n.06', 3.0, 1), ('act.n.02', 4.0, 303), ('knowledge_domain.n.01', 6.0, 332), ('analyze.v.01', 8.0, 141), ('idea
.n.01', 10.0, 0), ('artifact.n.01', 10.0, 336), ('work.v.01', 10.0, 313), ('work.v.02', 10.0, 314), ('show.n.03', 11.0, 165), ('move.v.02', 11.0, 79), ('no
te.v.01', 11.0, 31), ('succeed.v.01', 11.0, 315), ('act.v.02', 12.0, 305), ('state.v.01', 12.0, 42), ('worker.n.01', 12.0, 117), ('attribute.n.02', 12.0, 3
28), ('person.n.01', 12.0, 344), ('act.v.01', 12.0, 304), ('change.v.01', 12.0, 6), ('change.v.02', 13.0, 7), ('expert.n.01', 13.0, 178), ('happen.v.01', 1
3.0, 368), ('measure.n.02', 13.0, 200), ('collection.n.01', 13.0, 215), ('aid.n.02', 13.0, 254), ('good_person.n.01', 13.0, 275), ('game.n.01', 13.0, 302),
 ('occupation.n.01', 13.0, 309), ('think.v.01', 13.0, 162), ('material.n.01', 13.0, 338), ('ability.n.02', 14.0, 179), ('substance.n.01', 14.0, 335), ('hel
p.v.02', 15.0, 259), ('help.v.02', 15.0, 260), ('part.n.01', 15.0, 213), ('think.v.02', 15.0, 355), ('part.n.02', 16.0, 214), ('help.v.01', 16.0, 258), ('m
easure.v.04', 16.0, 186), ('result.n.03', 16.0, 364), ('touch.v.01', 16.0, 128), ('satisfy.v.02', 16.0, 221), ('change.n.01', 16.0, 5), ('quality.n.01', 16
.0, 329), ('imagine.v.01', 16.0, 3), ('accomplishment.n.01', 16.0, 363), ('abstract.a.01', 16.0, 2), ('failure.n.02', 16.0, 318), ('geological_formation.n.
01', 17.0, 135), ('travel.v.01', 17.0, 78), ('area.n.01', 17.0, 133), ('see.v.01', 17.0, 139), ('compound.n.02', 17.0, 341), ('number.n.02', 17.0, 216), ('
title.n.06', 17.0, 176), ('inform.v.01', 17.0, 44), ('information.n.01', 17.0, 43), ('manipulate.v.02', 17.0, 366), ('material.n.04', 18.0, 340), ('politic
s.n.03', 18.0, 171), ('organization.n.01', 18.0, 157), ('material.n.02', 18.0, 339), ('narrative.n.01', 18.0, 158), ('remark.n.01', 18.0, 30), ('aid.n.01',
 18.0, 253), ('insect.n.01', 18.0, 347), ('unpleasant_person.n.01', 18.0, 277), ('waste.n.01', 18.0, 287), ('ideal.s.02', 18.0, 4), ('remember.v.01', 18.0,
 48), ('learn.v.01', 18.0, 47), ('sound.n.01', 18.0, 34), ('location.n.01', 18.0, 132), ('sound.n.04', 18.0, 35), ('test.v.01', 18.0, 358), ('time_period.n
.01', 19.0, 92), ('touch.n.01', 19.0, 127), ('ownership.n.01', 19.0, 85), ('possession.n.02', 19.0, 86), ('develop.v.10', 19.0, 150), ('animal.n.01', 19.0,
 345), ('utility.n.02', 19.0, 255), ('property.n.04', 19.0, 327), ('resource.n.01', 19.0, 183), ('consumer_goods.n.01', 19.0, 352), ('equipment.n.01', 19.0
, 337), ('meet.v.02', 19.0, 29), ('confirm.v.01', 19.0, 359), ('show.v.04', 19.0, 166), ('resolve.v.06', 19.0, 14), ('rough.a.01', 20.0, 129), ('guess.v.02
', 20.0, 356), ('leader.n.01', 20.0, 173), ('affirm.v.02', 20.0, 360), ('structure.n.01', 20.0, 154), ('arrange.v.01', 20.0, 156), ('play.v.01', 20.0, 306)
, ('organ.n.01', 20.0, 234), ('psychological_state.n.01', 20.0, 331), ('food.n.02', 20.0, 101), ('medium_of_exchange.n.01', 20.0, 184), ('methodical.s.01',
 20.0, 18), ('journey.n.01', 20.0, 77), ('talk.v.01', 20.0, 40), ('talk.v.02', 20.0, 41), ('affirm.v.02', 20.0, 20)]

As a verb:
[]

As a descriptor:
[]
```