Project Proposal

1 Team Name and Members

The team name will be "Text-minder", essentially a slightly humorous way to express:

1. That the project will involve mining semantic information from text
2. That the goal is to allow users to tell a language model what it should say, i.e. the user babysits the textual output of the model.

I'm the only team member, name Jerry Ding, Andrew ID dalud.

2 The NL Task

The training objective of language models is the probability that the model generates text resembling the training data corpus, but language is used convey information so even an ideal language model can't necessarily predict one sentence from previous context. A language model can plausibly do well on tasks such as QA or document summarization, where useful information is available in the problem statement. But in free-form dialogue (such as daily conversation), an utterance can introduce new topics or information that can't be deduced from previous context. It seems to me that the act of choosing a response in a dialogue context has less to do with the statistical distribution of sentences in the language, and is more related to the speaker's knowledge of the world and the semantic content of the previous sentences.

The project works toward decoupling the process of generating natural-sounding language from the process of selecting the semantic content of a response. The goal is to create an generator engine whose responses are constrained to be relevant to externally-provided semantic information. This information will not be a complete representation of the desired output sentence. It instead only contains partial information about the people, objects and concepts that should be brought up in the desired output. This information should be simple enough that a human user can comfortably create the semantic constraints needed to define what he wants the language model to focus on.

When using a contemporary language model like GPT-2, the only method available for controlling its output is by adjusting previous text. This is very indirect and the model often ignores the commands, goals or information injected this way. For example, I have tried to use this method to inject information needed to disambiguate a syntactically ambiguous statement, but the injection success rate is very poor. This suggests that the language model isn't making interesting semantic inferences from the context, even though it uses word vectors which arguably contains some semantic information. This is why I think this task is a suitable use case for computational semantics.

The evaluation metric is the perplexity on a held-out data set, and the baselines would include an unconditioned decoder, and one that is only conditioned on the set of nouns in the sentence.

3 Source Materials, Data and Code

The data set I will be using to represent free-form dialogue will be the OpenSubtitles data set. The data set is unnecessarily big (~280M lines) so I am only taking a small subset (~3M lines). The lines come from movie subtitles, and from what I see in the data it seems unlikely that a character's line can be directly predicted from previous lines. I observe that contemporary language models quickly go off-topic when extrapolating new dialogue from this data.

Since natural-sounding language is still part of the project requirements, I will be using a pretrained GPT-2 model inside the system. Based on the computational resources I have, I believe the medium sized model (355M parameters) is a good fit. I don't plan to fine-tune the model with the dialogue data. Instead, I will add a new decoder on top that is designed to take in an encoding of the semantic constraints as well as hidden layer outputs of the pretrained language model. This "decoder head" is trained with the dialogue data and inferred constraints.

In my current vision of the semantic processing pipeline, I will need a relatively small set of primitives to describe actions and modifiers. I'm considering doing graph search through WordNet's semantic relation graph to associate words to primitive terms. In addition, I believe the best semantic inferences to use are probabilistic in nature, so I plan to use a Probabilistic Soft Logic engine to produce these inferences. I am thinking of using an existing parser for a shallow semantic representation (like TUPA) to assist in converting text into a format suitable for inference. I'll consider using Stanford's CoreNLP package to resolve co-references.

4 Semantic Model and System Architecture

4.1 The Semantic Constraints

The fundamental unit of semantic information that I intend to feed into the generator engine is the noun descriptor, instead of complete thoughts or composable tree structures. A noun descriptor encodes the constraint that the output sentence must mention a certain noun (possibly indirectly through co-reference), and that any attached description can inferred from the response or the previous context. The noun is discouraged from being an anaphor unless it is a first or second-person pronoun, and is otherwise not limited. The types of descriptors are:

- **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])`

To limit the complexity of the constraints, all descriptors and membership classes are chosen from a very limited set of primitive terms (100 - 1000 words).

4.2 Pipeline / Architecture

Firstly, the sentences are converted into a intermediate representation with relatively shallow semantic processing. Then a co-reference resolution engine is used to identify the nouns as specifically as possible. Next, a number of inference rules related to the small set of primitive terms will be created (may use both manual and automatic methods). In addition, the semantic relationship network from WordNet is used to relate words to the primitive terms when possible. The relations are processed with probabilistic soft logic to extract a list of semantic constraints from each sentence, from which a small subset are chosen. Here word sense disambiguation won't be done explicitly; the hope is that the inference engine can rule out word senses which result in invalid inferences, or that the impact of confused word senses is not too bad.

The extracted constraints are used as training data for the language model. For robustness, the constraints are randomly dropped out, and grouped by noun which will be randomly ordered.

5 Example Sentence and Representation

Taken randomly from the data set. Probability strengths of the constraints are omitted.

Sentences with context	Semantic Constraints

- `She's killed three already.`
- `It's gonna happen again soon.`
- `In Malaysia, if I had to guess.`
- `Malaysia?`
- **`I'm not sure what's going on here, but do you have proof of something?`**
	`Relevant(what)`
	`Action(I, -know[CMPLX, REAL])`
	`Action(I, +ask[You, REAL])`
	`Action(You, +own[Something, HYPO])`
	`Recipient(You, +ask[I, REAL])`
	`Recipient(Something, +own[You, HYPO])`
	`Member(Something, information)`

- `Not exactly, but how do you explain this?`
- `I used my friend Brad's workstation to do some digging around in the system.`
- **`Two days later, the gas line at Brad's house blew up, and he was burned to death.`**
	`Member(Day, time-duration)`
	`Member(Brad, owner)`
	`Member(Brad, friend)`
	`Member(Brad, male)`
	`Action(Brad, +burn[NONE, REAL])`
	`Descr(Brad, dead)`
	`Relevant(House)`
	`Action(Gas line, +damage[UNK, REAL])`

6 List of Modules

- The intermediate representation generator (a simplistic translation from the automatic UCCA representations created by TUPA)
- The co-reference resolution engine (directly from Stanford CoreNLP)
- Maybe a script to automatically propose semantic relationships across the primitive terms
- The semantic constraint generator (uses PSL with inference rules that I create)
- The pretrained GPT-2 model
- The decoder head (architecture will be based off Transformer)

7 Unknowns

I don't know much about what semantic representation parsers are readily available, and only saw TUPA from searching.

I'm not too sure what the primitive terms will be, though it will probably be derived from the most common English words.

Also I'm not too sure about the best way to use WordNet to link words to primitive terms.

Lastly I don't know about automatic ways to propose inference rules, or how long it would take to manually create enough rules to create interesting semantic constraints. Do I even need to do this, or are there readily available ways to generate inferences from text?