Shallow Discourse Parsing:
Discourse Relation Sense Classification Using Gold Argument Pairs

CS134 2017 Fall Final Project

Due Date: 12/15/2017

1 What is Shallow Discourse Parsing?

A typical text consists of sentences that are glued together in a systematic way to form a coherent discourse. Shallow discourse parsing is the task of parsing a piece of text into a set of discourse relations between two adjacent or non-adjacent discourse units. We call this task shallow discourse parsing because the relations in a text are not connected to one another to form a connected structure in the form of a tree or graph.

1.1 Discourse Analysis in the Penn Discourse Treebank

We use the Penn Discourse Treebank as the data set. The PDTB annotates a text with a set of discourse relations. A discourse relation is composed of:

- a discourse connective, which can be a coordinating conjunction (e.g., "and", "but"), subordinating conjunction (e.g. "if", "because"), or a discourse adverbial (e.g., "however", "also"). In an implicit discousre relaiton, a discourse connective is omitted.
- two Arguments of the discourse connective, Arg1 and Arg2, which are typically text spans the size of clauses or sentences.
- the sense of the discourse connective, which characterizes the nature of the relationship between the two arguments of the connective (e.g., contrast, instantiation, temporal precedence).

1.2 Examples of Discourse Relations

Here is a paragraph taken from the document wsj_1000 in the PDTB. A shallow discourse parser will output a bunch of discourse relations, which can be visualized below. Arg1 is shown in red, and Arg2 is shown in blue. The discourse connective is underlined.

Explicit Discourse Relations

> According to Lawrence Eckenfelder, a securities industry analyst at Prudential-Bache Securities Inc., "Kemper is the first firm to make a major statement with program trading." He added that "having just one firm do this isn't going to mean a hill of beans. But this prompts others to consider the same thing, then it may become much more important."
>
> The discourse connective is 'but', and the sense is Comparison.Concession.

Implicit Discourse Relations

> According to Lawrence Eckenfelder, a securities industry analyst at Prudential-Bache Securities Inc., "Kemper is the first firm to make a major statement with program trading." He added that "having just one firm do this isn't going to mean a hill of beans. But if this prompts others to consider the same thing, then it may become much more important."

The omitted discourse connective is 'however'. and the sense is Comparison.Contrast.

This project is modified from the CoNLL 2016 Shared Task. More information can be found here.

2 What to Do for this Final Project

For this final project, we are only doing discourse relation sense classification given gold argument pairs.

2.1 Data

We use Penn Discourse TreeBank (PDTB) 2.0, a 1-million-word Wall Street Journal corpus, for our experiments. The training data are from Sections 2-21, the development set is from Section 22, and Section 23 is used as the evaluation/test set.

relations.json

This file is from The Penn Discourse Treebank (PDTB), with gold standard annotation. Each line in the file is a json line. In Python, you can turn it into a dictionary. See Figure1 for an example.

```
In [4]: import json
        import codecs
        pdtb_file = codecs.open('conll16st-en-01-12-16-trial/relations.json', encoding='utf8')
        relations = [json.loads(x) for x in pdtb_file];
        example_relation = relations[10]
        example_relation
```


```
Out[4]: {u'Arg1': {u'CharacterSpanList': [[2493, 2517]],
          u'RawText': u'and told them to cool it',
          u'TokenList': [[2493, 2496, 465, 15, 8],
           [2497, 2501, 466, 15, 9],
           [2502, 2506, 467, 15, 10],
           [2507, 2509, 468, 15, 11],
           [2510, 2514, 469, 15, 12],
           [2515, 2517, 470, 15, 13]]},
         u'Arg2': {u'CharacterSpanList': [[2526, 2552]],
          u'RawText': u"they're ruining the market",
          u'TokenList': [[2526, 2530, 472, 15, 15],
           [2530, 2533, 473, 15, 16],
           [2534, 2541, 474, 15, 17],
           [2542, 2545, 475, 15, 18],
           [2546, 2552, 476, 15, 19]]},
         u'Connective': {u'CharacterSpanList': [[2518, 2525]],
          u'RawText': u'because',
          u'TokenList': [[2518, 2525, 471, 15, 14]]},
         u'DocID': u'wsj_1000',
         u'ID': 14887,
         u'Sense': [u'Contingency.Cause.Reason'],
         u'Type': u'Explicit'}
```


Figure 1: A json line in relations.json

The dictionary describes the following component of a relation:

- Arg1 : the text span of Arg1 of the relation
- Arg2 : the text span of Arg2 of the relation
- Connective : the text span of the connective of the relation
- DocID : document id where the relation is in.
- ID : the relation id, which is unique across training, dev, and test sets.

- Sense : the sense of the relation
- Type : the type of relation (Explicit, Implicit, Entrel, AltLex, or NoRel)

The text span is in the same format for Arg1, Arg2, and Connective. A text span has the following fields:

- CharacterSpanList : the list of character offsets (beginning, end) in the raw untokenized data file.
- RawText : the raw untokenized text of the span
- TokenList : the list of the addresses of the tokens in the form of (character offset begin, character offset end, token offset within the document, sentence offset, token offset within the sentence)

raw/wsj_****

These files contain the raw text data from PDTB, corresponding to character/token/sentence offsets in relations.json.

2.2 Project Requirements

1. Implement data processing code that (1) reads in training/develop/test data in the format as described in Section2.1, and (2) outputs sense classifications in the format as described in Section2.3.
2. Design and implement a Neural Network model for discourse relation sense classification.
3. Experiment with different hyper-parameters, feature representations, and/or neural network architectures.
4. Evaluate your system on explicit, implicit, and all relations, using the evaluation script described in Section2.4.
5. Write a project report:
   - Describe your feature representation, neural network architecture, and hyper-parameters design.
   - Describe your code structure, and how to run/test your system.
   - Describe your experiments, and report experiment results and analysis on the results.

2.3 What should the system output look like?

The system output must be in json format. It is very similar to the training set except for the TokenList field. The TokenList field is now a list of document level token indices. If the relation is not explicit, Connective field must still be there, and its TokenList must be an empty list. You may however add whatever field into json to help yourself debug or develop the system. Below is an example of a relation given by a system.

```
In [13]: import json
         import codecs
         output_relations = [json.loads(x) for x in codecs.open('output.json', encoding='utf8')]
         output_relations[0]

Out[13]: {'Arg1': {'TokenList': [275, 276, 277, 278, 279, 280, 281, 282]},
          'Arg2': {'TokenList': [329, 330, 331, 332, 333, 334, 335, 336, 337]},
          'Connective': {'TokenList': []},
          'DocID': 'wsj_1000',
          'Sense': ['Expansion.Conjunction'],
          'Type': 'Implicit'}
```

Figure 2: Output example.

2.4 Evaluation

Precision, recall, and F of the discourse relation sense classification are used as the evaluation metrics. To evaluate, run:

python scorer.py <path-to-gold-file>/relations.json <path-to-output-file>/output.json

Note: scorer.py, validator.py, and confusion_matrix.py are evaluation code that shouldn't be touched.