LING773 - Negotiations Project Report

Peng Ye, Youngil Kim, Olivia Buzek

May 3, 2011

1 Introduction

brief introduction Who is in this group. Optionally, a rough breakdown of people's roles in the project,

2 Preprocessing

3 Feature extraction

In this section, we will introduce three types of features that are extracted for predicting joint-profit.

3.1 Meta information

3.2 Code-based N-Gram feature

In our data set, every thought unit is carefully coded by social scientist. The code of a thought unit is like a summary about this thought unit. It helps us to understand the functionality of a thought unit. Some code indicates positive interaction between speakers, for example, 'RP' means positive reaction; some code indicates negative interaction, for example, 'RN' means negative reaction. Intuitively, negative or positive interaction should have high correlation with joint profit, thus thought unit code is informative feature for predicting joint profit. The second type of feature we use is though unit code based feature. Consider a dyad as a sequence of code, we then extract code-based unigram and bigram as our featurs. Specifically, for each dyad we extract the following features:

- Unigram based features:
  - Overall features: (Overall features are extracted from the entire dyad.) number of total thought units, number of thought units of each code, percentage of thought units of each code

- Wine features: (Wine features are extracted from wine part of the dyad)
  number of total thought units (in wine part), number of thought units of each code, percentage of thought units of each code
- Grocery features: (Grocery features are extracted from grocery part of the dyad)
  number of total thought units (in grocery part), number of thought units of each code, percentage of thought units of each annotation
- Relative feature: Relative feature indicates the difference between wine part and grocery part of the dyad.
  J-S divergence between the unigram distribution over wine dialog and unigram distribution over grocery dialog

- Bigram based features:
  - Overall features: (Overall features are extracted from the entire dyad.)
    number of turns (change of speakers) in the dyad, number of code-based bigram, percentage of code-based bigram
  - Wine features: (Wine features are extracted from wine part of the dyad)
    number of code-based bigram, percentage of code-based bigram
  - Grocery features: (Grocery features are extracted from grocery part of the dyad)
    number of code-based bigram, percentage of code-based bigram
  - Turning point features: A turning point in a dyad is where the speaker changes. Features extracted at turning points indicate how one speaker respond the other speaker.
    number of code-based bigram, percentage of code-based bigram
  - Relative feature:
    J-S divergence between the bigram distribution over wine dialog and bigram distribution over grocery dialog

In our corpus, we have 38 code-based unigram and 660 code-based bigram. We extract a total of 5515 code-based features from each dyad.

It is worth noting that features introduced above are redundant, for example the percentage of a unigram and the number of times a unigram occurred are correlated with each other and are only different by a constant factor, in the initial feature extraction part, we are not going to eliminate those redundant features, instead, we would like to collect as many features as possible and perform feature selection on these features to choose the most informative features. Feature selection would be illustrated in detail in next section.

3.3 Word-based N-Gram feature

We extract word-based n-grams through following process.

- Preprocessing with raw data - creating *fields_speaker.txt*:
  We analyzed *merged_turns.csv* and *metadata.csv* for creating *fields_speaker.txt* file. Because some dialogues do not have *joint profit* in *metadata.csv* file, we should rule out those dialogues even though they exist in *merged_turns.csv* file. In addition, we rearranaged sequence of fields for better looking; the dyid is positioned as the first item.

- Tokeninzing with whole dialog files:
  The raw data contains many special characters and not yet tokenized for creating n-grams. Basically, we used Treebank tokenization and this can be found in *http://www.cis.upenn.edu/∼treebank/tokenization.html*. Following list shows a part of the tokenization rules.

  - most punctuation is split from adjoining words.
  - verb contractions and the Anglo-Saxon genitive of nouns are split into their component morphemes, and each morpheme is tagged separately.
    * Examples
      · children's →children 's
      · I'm →I 'm

    This tokenization allows us to analyze each component separately.
    In addition, we need to correct some words to see more precise result. For example, some speakers omit *question mark* or *apostrophe*, and sometimes there are several special characters which is not interpretable in plain text. You can see the detailed conversion in *tokenize* function in *create_ngram.py*.

- Count n-grams and Create n-gram ID:
  After tokenizing, we count number of n-grams to see the characteristics of each dialogue. For counting ngrams, we used *count.pl* in Ted Pedersen Ngram Statics Package with general stop words because stop words can hide real characteristics of n-grams. We seperately count n-grams for Wine, Grocery, and all dialogues for seeing the characteristics of each group of speakers. Also, we count n-grams per each dyad too.
  In addition, we created n-gram id files to express ngrams. For example, *grand opening date* which is one of trigram is expressed as *3-0001*. By using this ngram id, we can save memory and storage for analyzing whole data.

- Make n-gram feature file:
  By using counted n-grams, we created n-gram feature file which can be converted into an ARFF format, input format of Weka. Each row in the file contains n-gram information of each dyad. The first column shows the *dyID*, and the following

colums show ngram id and the number of ngrams in the dialogues. Following example shows a part of the feature file.

- Example

```
02,1-0002:30,1-0001:23,1-0040:18,1-0009:16,1-0012:15 ... ...
03,1-0006:30,1-0001:29,1-0048:23,1-0026:21,1-0025:20 ... ...
```

We made three n-gram feature files for each category, Wine, Grocery, and all dialogues.

In our experiment, we actually used only the n-gram extracted from entire dyad only. We have a total of 5754 word-based ngram features.

4 Regression

4.1 Feature Selection

Feature subset selection is a process of identifying the most informative features and removing irrelevant and redundant features. In Weka, we use 'BestFirst' as search method and use 'CfsSubsetEval' as attribute evaluator for finding the most informative features. As is explained in Weka, 'CfsSubsetEval' evaluates "the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them" and 'BestFirst' searches "the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility. Setting the number of consecutive non-improving nodes allowed controls the level of backtracking done." In our experiment, best first search start with the full set of attributes and search backward. The top ten best word based ngram feature is shown in Table.1 and the top ten best code based ngram feature is shown in Table. In this table, "percentage-of-(MIN-QR)-turns" means the percentage of the bigram of MIN-QR extracted from turning points of a dyad.

We extract the top 20 most informative code-based features and word-based feature, together with features extracted from meta information as the final feature. Specifically, we have a total of 49 features for each of the 61 dyads. It should be noticed that feature selection performed on all the available data may lead to overfitting problem, however, we would like to first perform feature selection on all data to see which features are the best for our current dataset. In the following experimental part, we will split dataset into training and testing part and feature selection would be only performed on training data to make sure we won't suffer overfitting problem.

4.2 Regression Algorithms

We used three different types of regression methods provided in Weka for predicting joint profit.

Table 1: THE TOP TEN MOST INFORMATIVE WORD-BASED AND CODE-BASED NGRAM FEATURE

CODE-BASED NGRAM FEATURE	WORD-BASED NGRAM
percentage-of-(MIN-QR)-turns	hours
percentage-of-(SF)-overall	special
percentage-of-(SF)-grocery	great
number-of-(IR)-overall	30k
percentage-of-(SBR-SF)-overall	agreeable
percentage-of-(OM-QM)-wine	630
percentage-of-(MIN-IR)-wine	1030pm
percentage-of-(IP-SF)-grocery	luxurious
percentage-of-(IP-IDN)-turn	people work
percentage-of-(IDN-SBR)-turns	crust grocery

- weka.classifiers.functions.SMOreg: support vector machine for regression, and we use RBF kernel.
- weka.classifiers.functions.LinearRegression: using linear regression for prediction.
- weka.classifiers.meta.Bagging + weka.classifiers.trees.REPTree: bagging predictor based on REPTree which builds a regression tree using information gain.

4.3 Evaluation Metrics

The following metrics are used to evaluate the proposed method.

1. Correlation coefficient (CC)
2. Mean absolute error (MAE)
3. Root mean squared error (RMSE)
4. Relative absolute error (RAE)
5. Root relative squared error (RRSE)

5 Experiment results

In this section, we will show the experimental results using the features obtained through the feature selection process and the three different regression methods. This result is compared to a baseline experiment, where code-based unigram features and meta information are directly used to form a 241-by-1 feature vector (without further feature selection process) for each dyad. Our experimental results show that using those selected features, we can achieve better performance.

Table 2: EXPERIMENT RESULTS WITH TEN FOLD CROSS-VALIDATION AND FEATURE SELECTION PERFORMED ON ALL DATA

		CC	MAE	RMSE	RAE	RRSE
SVM	Baseline	0.72	57.93	74.01	64.90%	70.44%
	Improved	0.90	43.03	53.25	48.20%	50.68%
Linear	Baseline	0.71	64.32	86.42	72.05%	82.25%
Regression	Improved	0.76	69.30	88.11	77.63%	83.87%
REPTree	Baseline	0.66	67.75	81.42	75.89%	77.50%
Bagging	Improved	0.69	64.16	77.01	71.87%	73.49%

Table 3: EXPERIMENT RESULTS WITH FIVE FOLD CROSS-VALIDATION AND FEATURE SELECTION PERFORMED ON ALL DATA

		CC	MAE	RMSE	RAE	RRSE
SVM	Baseline	0.68	66.33	83.57	74.15%	79.63%
	Improved	0.89	44.31	54.41	49.53%	51.85%
Linear	Baseline	0.67	70.47	90.95	78.77%	86.67%
Regression	Improved	0.53	82.82	105.08	92.58%	100.13%
REPTree	Baseline	0.51	73.06	89.84	81.67%	85.61%
Bagging	Improved	0.69	62.44	78.01	69.80%	74.34%