TRIVENI ASHOK NAIK
SUID: 406461644

NLP PROJECT

Option-1 - sentiment classification experiment + design a chatbot

Sentiment classification

I have performed three experiments in the NLTK on the movie review sentence data in which I have compared the baseline performance of using just word features with some modified or additional features that's been implemented. In the three experiments preformed, I have

- Varied the representation of the subjectivity lexicon features .
- Combine POS tag counts and subjectivity lexicon features.
- Combine bigram counts and subjectivity lexicon features.

Experiment 1: Varied the representation of the subjectivity lexicon features .

In [66]:
```
# define features that include word counts of subjectivity words
# negative feature will have number of weakly negative words +
#    2 * number of strongly negative words
# positive feature has similar definition
#    not counting neutral words
def SL_features(document, word_features, SL):
    document_words = set(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive' and posTag == 'verb':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive' and posTag == 'noun':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
            features['positivecount'] = weakPos + (2 * strongPos)
            features['negativecount'] = weakNeg + (2 * strongNeg)
    return features
```

Here for this I have created a feature function, as it is done in the class. This function has all the word features as before, but also has two features 'positivecount' and 'negativecount'. These features contains counts of all the positive and negative subjectivity words, where each weakly subjective word is counted once and each strongly subjective word is counted twice.

TRIVENI ASHOK NAIK
SUID: 406461644

Module Subjectivity reads the subjectivity lexicon file.

The data structure that is created is a dictionary where each word is mapped to a list of 4 things:

- strength, which will be either 'strongsubj' or 'weaksubj'
- posTag, either 'adj', 'verb', 'noun', 'adverb', 'anypos'
- isStemmed, either true or false
- polarity, either 'positive', 'negative', or 'neutral'

I have used a combination of these thing to determine the accuracy, i.e. the strength, posTag and the polarity.

I have taken posTag to be named as ‘verb’ and ‘noun’.

Later, I have created the feature sets but using the feature extraction function.

```
In [67]: SL_featuresets = [(SL_features(d, word_features, SL), c) for (d, c) in documents]

         # show just the two sentiment lexicon features in document 0
         print(SL_featuresets[0][0]['positivecount'])
         print(SL_featuresets[0][0]['negativecount'])

         0
         2

In [68]: # this gives the label of document 0
         SL_featuresets[0][1]
         # number of features for document 0
         len(SL_featuresets[0][0].keys())

Out[68]: 2002

In [69]: # retrain the classifier using these features
         train_set, test_set = SL_featuresets[1000:], SL_featuresets[:1000]
         classifier = nltk.NaiveBayesClassifier.train(train_set)
         nltk.classify.accuracy(classifier, test_set)

Out[69]: 0.743
```

- In my random training, test split, these particular sentiment features did improve the classification on this dataset.

- Now , for cross-validation different chunks of the data is used as the test set to repeatedly train a model and then average our performance over those models.
- This method is called cross-validation, or sometimes k-fold cross-validation. In this method, we choose a number of folds, k, which is usually a small number like 5 or 10.

  I have used the fold of 5, for all the methods here. The mean accuracy is also generated here.

In [72]:
```
## cross-validation ##
# this function takes the number of folds, the feature sets
# it iterates over the folds, using different sections for training and testing in turn
#   it prints the accuracy for each fold and the average accuracy at the end
def cross_validation_accuracy(num_folds, featuresets):
    subset_size = int(len(featuresets)/num_folds)
    accuracy_list = []
    # iterate over the folds
    for i in range(num_folds):
        test_this_round = featuresets[(i*subset_size):][:subset_size]
        train_this_round = featuresets[:(i*subset_size)] + featuresets[((i+1)*subset_size):]
        # train using train_this_round
        classifier = nltk.NaiveBayesClassifier.train(train_this_round)
        # evaluate against test_this_round and save accuracy
        accuracy_this_round = nltk.classify.accuracy(classifier, test_this_round)
        print (i, accuracy_this_round)
        accuracy_list.append(accuracy_this_round)
    # find mean accuracy over all rounds
    print ('mean accuracy', sum(accuracy_list) / num_folds)
```

In [71]:
```
# perform the cross-validation on the featuresets with word features and generate accuracy
num_folds = 5
cross_validation_accuracy(num_folds, featuresets)
```

```
0 0.7429643527204502
1 0.7462476547842402
2 0.7457786116322702
3 0.7514071294559099
4 0.7396810506566605
mean accuracy 0.7452157598499062
```

Results: After training an accuracy of 0.743 is obtained and after performing cross validation for 5 folds, an mean accuracy of 0.7452 is obtained.

Experiment 2: Combine POS tag counts and subjectivity lexicon features.

For this experiment I have combined the POS tag counts and the subjectivity lexicon features. There are some classification tasks where part-of-speech tag features can have an effect . The most common way to use POS tagging information is to include counts of various types of word tags. The below function has an example feature function that counts nouns, verbs, adjectives and adverbs for features. I have combined this feature with the subjectivity lexicon feature and hence I have determined the accuracy for this.

Later I have also performed cross validation for this for 5 fold and have determined the accuracies.

Screenshots for these is given below.

```
    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive' and posTag == 'verb':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive' and posTag == 'noun':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
            features['positivecount'] = weakPos + (2 * strongPos)
            features['negativecount'] = weakNeg + (2 * strongNeg)

    for (word, tag) in tagged_words:
        if tag.startswith('N'): numNoun += 1
        if tag.startswith('V'): numVerb += 1
        if tag.startswith('J'): numAdj += 1
        if tag.startswith('R'): numAdverb += 1
    features['nouns'] = numNoun
    features['verbs'] = numVerb
    features['adjectives'] = numAdj
    features['adverbs'] = numAdverb
    return features
```

In [46]:
```
SL_featuresets = [(SL_features(d, word_features, SL), c) for (d, c) in documents]

# show just the two sentiment lexicon features in document 0
print(SL_featuresets[0][0]['positivecount'])
print(SL_featuresets[0][0]['negativecount'])

# define feature sets using this function
#POS_featuresets = [(SL_features(d, word_features, 0), c) for (d, c) in documents]
# number of features for document 0
print(len(SL_featuresets[0][0].keys()))
```

```
0
1
2006
```

In [47]:
```
# this gives the label of document 0
SL_featuresets[0][1]
# number of features for document 0
len(SL_featuresets[0][0].keys())
# the first sentence
print(documents[0])

# the pos tag features for this sentence
print('num nouns', SL_featuresets[0][0]['nouns'])
print('num verbs', SL_featuresets[0][0]['verbs'])
print('num adjectives', SL_featuresets[0][0]['adjectives'])
print('num adverbs', SL_featuresets[0][0]['adverbs'])
```

```
(['barely', 'goes', 'beyond', 'comic', 'book', 'status', '.'], 'neg')
num nouns 2
num verbs 1
num adjectives 1
num adverbs 1
```

In [48]:
```
# retrain the classifier using these features
train_set, test_set = SL_featuresets[1000:], SL_featuresets[:1000]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

Out[48]: 0.774

In [49]:
```
## cross-validation ##
# this function takes the number of folds, the feature sets
# it iterates over the folds, using different sections for training and testing in turn
#   it prints the accuracy for each fold and the average accuracy at the end
def cross_validation_accuracy(num_folds, featuresets):
    subset_size = int(len(featuresets)/num_folds)
    accuracy_list = []
    # iterate over the folds
    for i in range(num_folds):
        test_this_round = featuresets[(i*subset_size):][:subset_size]
        train_this_round = featuresets[:(i*subset_size)] + featuresets[((i+1)*subset_size):]
        # train using train_this_round
        classifier = nltk.NaiveBayesClassifier.train(train_this_round)
        # evaluate against test_this_round and save accuracy
        accuracy_this_round = nltk.classify.accuracy(classifier, test_this_round)
        print (i, accuracy_this_round)
        accuracy_list.append(accuracy_this_round)
    # find mean accuracy over all rounds
    print ('mean accuracy', sum(accuracy_list) / num_folds)
```

In [50]:
```
# perform the cross-validation on the featuresets with word features and generate accuracy
num_folds = 5
cross_validation_accuracy(num_folds, featuresets)
```

```
0 0.7453095684803002
1 0.7354596622889306
2 0.7476547842401501
3 0.7476547842401501
4 0.75
mean accuracy 0.7452157598499062
```

Results :

After training an accuracy of 0.774 is obtained and after performing cross validation for 5 folds, an mean accuracy of 0.7452 is obtained.

Experiment 3: Combine bigram counts and subjectivity lexicon features.

For this experiment I have combined the bigram counts and the subjectivity lexicon features.One more important source of features often used in sentiment and other document or sentence-level classifications is bigram features. We do the similar thing initially like by loading the movie review sentences and getting the baseline performance of the unigram features.

The screenshot below depicts the combination of two features.

TRIVENI ASHOK NAIK
SUID: 406461644

In [32]:
```
# define features that include words as before
# add the most frequent significant bigrams
# this function takes the list of words in a document as an argument and returns a feature dictionary
# it depends on the variables word_features and bigram_features
def bigram_document_features(document, word_features, bigram_features):
    document_words = set(document)
    document_bigrams = nltk.bigrams(document)
    tagged_words = nltk.pos_tag(document)
    features = {}
    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)

    numNoun = 0
    numVerb = 0
    numAdj = 0
    numAdverb = 0

    for word in word_features:
        features['contains({})'.format(word)] = (word in document_words)
    # count variables for the 4 classes of subjectivity
    weakPos = 0
    strongPos = 0
    weakNeg = 0
    strongNeg = 0

    for word in document_words:
        if word in SL:
            strength, posTag, isStemmed, polarity = SL[word]
            if strength == 'weaksubj' and polarity == 'positive' and posTag == 'verb':
                weakPos += 1
            if strength == 'strongsubj' and polarity == 'positive' and posTag == 'noun':
                strongPos += 1
            if strength == 'weaksubj' and polarity == 'negative':
                weakNeg += 1
            if strength == 'strongsubj' and polarity == 'negative':
                strongNeg += 1
            features['positivecount'] = weakPos + (2 * strongPos)
            features['negativecount'] = weakNeg + (2 * strongNeg)
```

```
    for (word, tag) in tagged_words:
        if tag.startswith('N'): numNoun += 1
        if tag.startswith('V'): numVerb += 1
        if tag.startswith('J'): numAdj += 1
        if tag.startswith('R'): numAdverb += 1
    features['nouns'] = numNoun
    features['verbs'] = numVerb
    features['adjectives'] = numAdj
    features['adverbs'] = numAdverb

    for bigram in bigram_features:
        features['bigram({} {})'.format(bigram[0], bigram[1])] = (bigram in document_bigrams)
    return features
```

The function to generate the bigram_featuresets id given below and also the total number of bigram feature _sets is also determined.

In [33]:
```
# use this function to create feature sets for all sentences
bigram_featuresets = [(bigram_document_features(d, word_features, bigram_features), c) for (d, c) in documents]

# number of features for document 0
print(len(bigram_featuresets[0][0].keys()))

# features in document 0
print(bigram_featuresets[0][0])

print(len(bigram_featuresets[0][0].keys()))
```

```
2506
{'contains(.)': True, 'contains(the)': True, 'contains(,)': True, 'contains(a)': False, 'contains(and)': False, 'cont
ains(of)': True, 'contains(to)': False, 'contains(is)': False, 'contains(in)': True, 'contains(that)': False, 'contai
ns(it)': False, 'contains(as)': False, 'contains(but)': False, 'contains(with)': True, 'contains(film)': False, 'cont
ains(this)': False, 'contains(for)': False, 'contains(its)': False, 'contains(an)': False, 'contains(movie)': True,
"contains(it's)": False, 'contains(be)': True, 'contains(on)': False, 'contains(you)': False, 'contains(not)': False,
'contains(by)': False, 'contains(about)': False, 'contains(more)': False, 'contains(one)': False, 'contains(like)': F
alse, 'contains(has)': False, 'contains(are)': False, 'contains(at)': True, 'contains(from)': False, 'contains(tha
n)': False, 'contains(")': False, 'contains(all)': False, 'contains(--)': False, 'contains(his)': True, 'contains(hav
e)': False, 'contains(so)': False, 'contains(if)': True, 'contains(or)': False, 'contains(story)': False, 'contains
(i)': False, 'contains(too)': True, 'contains(just)': False, 'contains(who)': False, 'contains(into)': False, 'contai
ns(what)': False, 'contains(most)': False, 'contains(out)': False, 'contains(no)': False, 'contains(much)': True, 'co
ntains(even)': False, 'contains(good)': False, 'contains(up)': True, 'contains(will)': False, 'contains(comedy)': Fal
se, 'contains(time)': False, 'contains(can)': False, 'contains(some)': False, 'contains(characters)': False, 'contain
s(only)': False, 'contains(little)': False, 'contains(way)': True, 'contains(their)': False, 'contains(funny)': Fals
e, 'contains(make)': False, 'contains(enough)': False, 'contains(been)': False, 'contains(very)': False, 'contains(yo
ur)': False, 'contains(never)': False, 'contains(when)': False, 'contains(makes)': False, 'contains(there)': False,
'contains(may)': False, 'contains(which)': False, 'contains(us)': False, 'contains(work)': False, 'contains(best)': F
alse, 'contains(he)': True, 'contains(bad)': False, "contains(doesn't)": False, 'contains(director)': False, 'contain
```

The accuracy got after training the bigram feature sets is as given below. To determine the number of nouns, verbs, adjectives and adverbs present, bigram feature_sets function is used.

In [34]:
```
# the first sentence
print(documents[0])
# the pos tag features for this sentence
print('num nouns', bigram_featuresets[0][0]['nouns'])
print('num verbs', bigram_featuresets[0][0]['verbs'])
print('num adjectives', bigram_featuresets[0][0]['adjectives'])
print('num adverbs', bigram_featuresets[0][0]['adverbs'])
```

```
(['if', 'welles', 'was', 'unhappy', 'at', 'the', 'prospect', 'of', 'the', 'human', 'race', 'splitting', 'in', 'two',
',', 'he', 'probably', "wouldn't", 'be', 'too', 'crazy', 'with', 'his', "great-grandson's", 'movie', 'splitting', 'u
p', 'in', 'pretty', 'much', 'the', 'same', 'way', '.'], 'neg')
num nouns 6
num verbs 4
num adjectives 6
num adverbs 4
```

In [35]:
```
# train a classifier and report accuracy
train_set, test_set = bigram_featuresets[1000:], bigram_featuresets[:1000]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

Out[35]: 0.73

CrossValidation is performed and it take as the same steps as done before for th epos tag counts and hence it is done for 5 folds . The accuracy is determined.

In [36]:
```
## cross-validation ##
# this function takes the number of folds, the feature sets
# it iterates over the folds, using different sections for training and testing in turn
#   it prints the accuracy for each fold and the average accuracy at the end
def cross_validation_accuracy(num_folds, featuresets):
    subset_size = int(len(featuresets)/num_folds)
    accuracy_list = []
    # iterate over the folds
    for i in range(num_folds):
        test_this_round = featuresets[(i*subset_size):][:subset_size]
        train_this_round = featuresets[:(i*subset_size)] + featuresets[((i+1)*subset_size):]
        # train using train_this_round
        classifier = nltk.NaiveBayesClassifier.train(train_this_round)
        # evaluate against test_this_round and save accuracy
        accuracy_this_round = nltk.classify.accuracy(classifier, test_this_round)
        print (i, accuracy_this_round)
        accuracy_list.append(accuracy_this_round)
    # find mean accuracy over all rounds
    print ('mean accuracy', sum(accuracy_list) / num_folds)
```

In [37]:
```
# perform the cross-validation on the featuresets with word features and generate accuracy
num_folds = 5
cross_validation_accuracy(num_folds, featuresets)
```

```
0 0.7326454033771107
1 0.7448405253283302
2 0.7439024390243902
3 0.7603189493433395
4 0.7378048780487805
mean accuracy 0.7439024390243902
```

Results: After training an accuracy of 0.73 is obtained and after performing cross validation for 5 folds, an mean accuracy of 0.7439 is obtained.

ChatBot

I have developed an Chatbot called as the FoodOrderAgent. This system will help the users to place an order of whatever they wish to eat from their favorite restaurants. And also apart from this it also helps in monitoring how long the food will take to get delivered.

I have two intents

- Place an order
- Estimated time delivery

Intent 1: Place an order

For the first intent i.e. the place an order intent, I have 7 utterances with 6 entities.
And also for the second intent i.e., the estimated time delivery intent, I have 5 utterances with 5 entities.

The utterances for the place an order intent is

1. Place an order of 3 cheese pizza at www.papajohns.com
2. Place an order of chicken nuggets from www.burgerking.com which costs around $3.14
3. Place an order of 3 extra large cheese fries from dennys.
4. Place an order of 3 iced white chocolate mocha from Starbucks which costs around $4.13 each.

TRIVENI ASHOK NAIK
SUID: 406461644

5. Place an order of 4 french toasts from PapaJohns and send the receipt to naiktriveni@gmail.com
6. Place an order of nine cappuccinos from StarBucks.
7. Place an order of an Chinese noodles with 10 % less oil.

The entities that I have used here are

- Number
- url
- money
- email
- percentage

Utterance 1: Place an order of 3 cheese pizza at www.papajohns.com

For this sentence, since the entity number and url is present, 3 takes in as number and www.papajohns.com is taken as url.

Utterance 2: Place an order of chicken nuggets from www.burgerking.com which costs around $3.14

For this sentence, since the entity money and url is present, $3.14 takes is taken as money and since the number entity is also present it takes 3.14 as number and www.burgerking is taken as url.

Utterance 3: Place an order of 3 extra large cheese fries from dennys.

For this sentence, since the entity number is present, 3 takes in as number .

Utterance 4: Place an order of 3 iced white chocolate mocha from Starbucks which costs around $4.13 each.

For this sentence, since the entity money and number is present, $4.13 is taken as money and since the number entity is also present it takes 4.13 as number and 3 is taken as number.

Utterance 5: Place an order of 4 french toasts from PapaJohns and send the receipt to naiktriveni@gmail.com

For this sentence, since the entity email and number is present, 4 is taken as number and since the email entity is also present it takes naiktriveni@gmail.com as email.

Utterance 6: Place an order of nine cappuccinos from StarBucks.

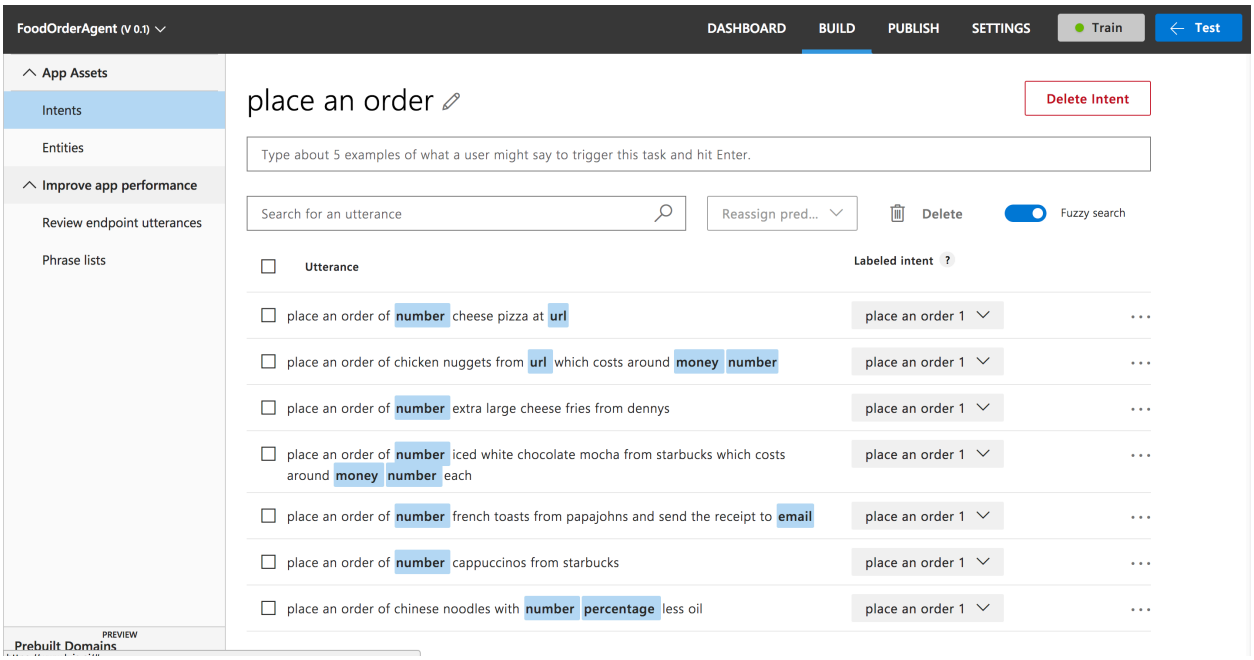
For this sentence, since the number is present, nine is taken as number.

Utterance 7: Place an order of an Chinese noodles with 10 % less oil.

TRIVENI ASHOK NAIK
SUID: 406461644

For this sentence, since the entity percentage and number is present, 10 is taken as number and since the percentage entity is also present it takes 10% as percentage.

Some of the screenshots of the application is



After Training and testing and later on publishing, I tried out some examples here are some.
Query 1 : Place an order of 10 bacon pizza from www.californiapizza.com

```
{
  "query": " place an order of 10 bacon pizza from www.californiapizza.com",
  "topScoringIntent": {
    "intent": "place an order",
    "score": 0.999994755
  },
  "intents": [
    {
      "intent": "place an order",
      "score": 0.999994755
    },
    {
      "intent": "None",
      "score": 0.04285432
    },
    {
      "intent": "estimated delivery",
      "score": 7.48420064E-08
    }
  ],
  "entities": [
    {
      "entity": "www.californiapizza.com",
      "type": "builtin.url",
      "startIndex": 39,
      "endIndex": 61
    },
    {
      "entity": "10",
      "type": "builtin.number",
      "startIndex": 19,
      "endIndex": 20,
      "resolution": {
        "value": "10"
      }
    }
  ]
}
```


Query 2: Place an order of 3 coffees from CaféNoir and send the receipt to trnaik@syr.edu


```
{
  "query": "place an order of 3 coffes from cafenoir and send the receipt to trnaik@syr.edu",
  "topScoringIntent": {
    "intent": "place an order",
    "score": 0.9997673
  },
  "intents": [
    {
      "intent": "place an order",
      "score": 0.9997673
    },
    {
      "intent": "None",
      "score": 0.0360708
    },
    {
      "intent": "estimated delivery",
      "score": 5.498748E-07
    }
  ],
  "entities": [
    {
      "entity": "trnaik@syr.edu",
      "type": "builtin.email",
      "startIndex": 65,
      "endIndex": 78
    },
    {
      "entity": "3",
      "type": "builtin.number",
      "startIndex": 18,
      "endIndex": 18,
      "resolution": {
        "value": "3"
      }
    }
  ]
}
```

TRIVENI ASHOK NAIK
SUID: 406461644

Intent 2 : Estimated delivery

The utterances for the place an order intent is

1. Tell me the time when the 3 pizzas ordered at www.papajohns.com will get delivered.
2. What is the estimated delivery of the rice bowl to Avondale which is 2miles away.
3. What is the estimated delivery time of the 2 medium size pizza to Avondale.
4. Tell me the delivery time of all the food items delivered at www.papajohns.com
5. What is the estimated delivery of 3 items ordered which costs $4.16

The entities that I have used here are

- Location
- url
- number
- money
- dimension

Utterance 1: Tell me the time when the 3 pizzas ordered at www.papajohns.com will get delivered.

For this sentence, since the entity url and number is present, 3 is taken as number and since the url entity is also present it takes www.papajohns.com as url.

Utterance 2: What is the estimated delivery of the rice bowl to Avondale which is 2miles away.

For this sentence, since the entity dimension and location is present, 2miles is taken as dimension and since the location entity is also present it takes Avondale as location.

Utterance 3: What is the estimated delivery time of the 2 medium size pizza to Avondale.

For this sentence, since the entity number and location is present, 2 is taken as number and since the location entity is also present it takes Avondale as location.

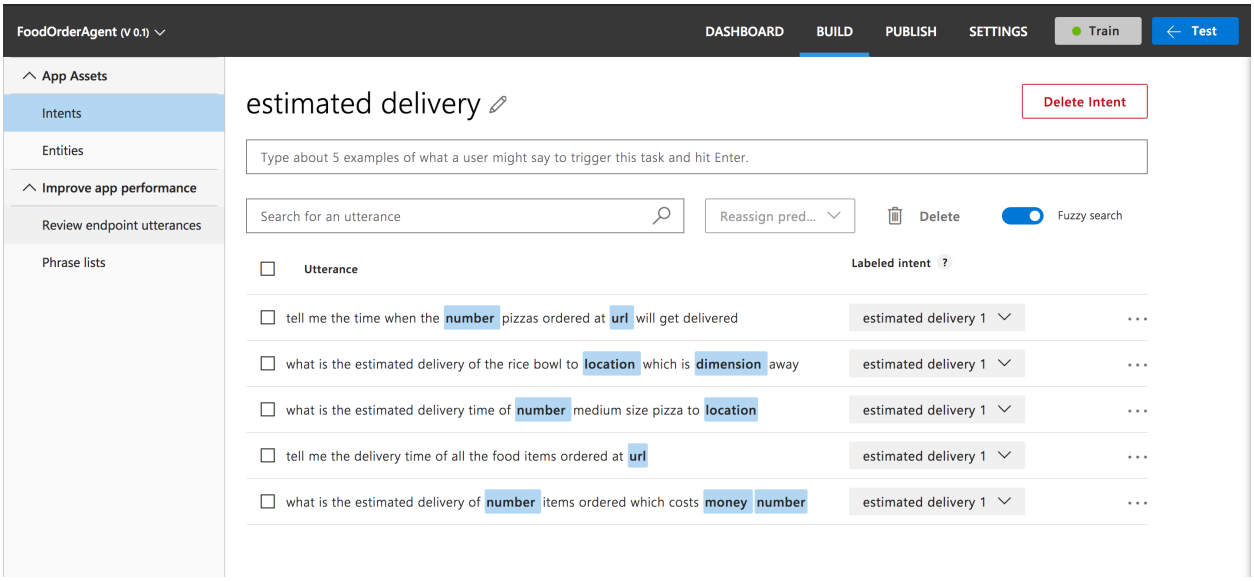
Utterance 4: Tell me the delivery time of all the food items delivered at www.papajohns.com

For this sentence, since the entity url is present, www.papajohns.com is taken as url .

Utterance 5: What is the estimated delivery of 3 items ordered which costs $4.16

For this sentence, since the entity money and number is present, $4.16 is taken as money and since the number entity is also present it takes 4.16 as number and 3 is taken as number.

TRIVENI ASHOK NAIK
SUID: 406461644



Some of the examples that I have tried, is given below

Query : what is the estimated time required to deliver 6 coffees to avondale from www.starbucks.com


```
{
  "query": " what is the estimated time required to deliver 6 coffees to avondale from www.starbucks.com",
  "topScoringIntent": {
    "intent": "estimated delivery",
    "score": 0.9980878
  },
  "intents": [
    {
      "intent": "estimated delivery",
      "score": 0.9980878
    },
    {
      "intent": "None",
      "score": 0.0278410632
    },
    {
      "intent": "place an order",
      "score": 0.000802380731
    }
  ],
  "entities": [
    {
      "entity": "www.starbucks.com",
      "type": "builtin.url",
      "startIndex": 75,
      "endIndex": 91
    },
    {
      "entity": "6",
      "type": "builtin.number",
      "startIndex": 48,
      "endIndex": 48,
      "resolution": {
        "value": "6"
      }
    }
  ]
}
```


URL : https://westus.api.cognitive.microsoft.com/luis/v2.0/apps/adaa935c-4a47-455e-9ac6-164c7b6963e8?subscription-key=d75e37298c03477cbad1f74bf2bca1c7&spellCheck=true&bing-spell-check-subscription-key={YOUR_BING_KEY_HERE}&verbose=true&timezoneOffset=0&q=

TRIVENI ASHOK NAIK
SUID: 406461644