NLP First Assignment
Jagat Sastry Pudipeddi (SB ID 108721027)

Task 1

Performance, based on the features given in the paper (Task 3 contains a table of 3 more new features)

		Truthful			Deceptive		
Features	Accuracy	P	R	F	P	R	F
POS_{SVM}	73%	80.07%	61.25%	69.41%	68.62%	84.75%	75.84%
$UNIGRAMS_{SVM}$	89.75%	**90.77%**	88.50%	89.62%	88.78%	**91.00%**	**89.88%**
$BIGRAMS^{+}_{SVM}$	90.00%	88.83%	91.50%	80.15%	91.24%	88.50%	89.85%
$TRIGRAMS^{+}_{SVM}$	**90.38%**	88.36%	93.00%	**90.62%**	92.61%	87.75%	90.12%
$UNIGRAMS_{NB}$	83.12%	76.55%	**95.50%**	84.98%	**94.02%**	70.75%	80.74%
$BIGRAMS^{+}_{NB}$	88.62%	84.56%	94.50%	89.26%	93.77%	82.75%	87.92%
$TRIGRAMS^{+}_{SVM}$	90.00%	87.38%	93.50%	90.34%	93.01%	86.50%	89.64%

Table: Automated classifier performance for the approaches based on nested 5-fold cross-validation experiments. Reported precision, recall and F-score are computed using a micro-average, i.e., from the aggregate true positive, false positive and false negative rates, as suggested by Forman and Scholz (2009).

Languages and tools used
Python: For programming the whole task
NLTK: For tokenizing, generating N-Grams and POS tags and calculating Frequency distribution. Also for performing naïve Bayes classification
SVMLight: Since the built-in SVM provided with NLTK didn't scale enough for the task, svmlight tool was used. svm_learn and svm_classify were called as external processes
Bash: For driving the n-gram SVM based classification process (calling individual python scripts)

Description of the implementation of each classification process
I wrote generic code for all N-Gram task so that given a particular natural number n, it generates 1-gram, 2-grams....n-grams and performs the classification based on n-gram+ using either SVMLight or NLTK-NB, depending on the task at hand.

SVM (ngram and POS)
To be called as
./project_driver.sh $N
Where $N represents the number *n* in *n-gram.* For POSsvm, let $N be 0

The project driver generates the training and test files for nested cross-validation, runs nested cross-validation, selects the best C value and tests that fold with that C. It generates report and displays the accuracy, precision, recall and f-score for each fold as well as the micro-average report for that run.

NLP First Assignment
Jagat Sastry Pudipeddi (SB ID 108721027)

Feature sets used

For a given N-Gram sequence (E.g. a pair of tokens in case of bigram), the ***normalized TF-IDF*** value is used as the feature value and the given n-gram sequence is the feature. The normalized TF-IDF is calculated as

Freq[ngram]/maxFd * math.log(TOT_NUM_DOCS/df[ngram])

where ngram is the ngram sequence, maxFd is the maximum frequency of any ngram sequence in that document, TOT_NUM_DOCS is the total number of docs and df[ngram] is the number of documents containing that ngram sequence.

For POS based classification, each POS is considered as a unigram feature and the same algorithm that was used for $\text{UNIGRAM}_{\textbf{SVM}}$ is used.

The best value for parameter C that were generated by the nested cross-validation code are

Unigram	Bigram	Trigram
Fold 0 : 0.0 Fold 1 : 0.0 Fold 2 : 0.0 Fold 3 : 0.1 Fold 4 : 0.0	Fold 0 : 0.1 Fold 1 : 0.1 Fold 2 : 0.0 Fold 3 : 0.0 Fold 4 : 0.1	Fold 0 : 0.1 Fold 1 : 0.1 Fold 2 : 0.1 Fold 3 : 0.1 Fold 4 : 0.1

Note: These values can be obtained by running the following command after having run the project driver

grep -o "For fold . : ...." svm_in_out/svmlight_output_$N_gram

Naive Bayes Classifier

To be invoked as

python nb_reader $N

where $N is as was used for SVM. The program performs the 5-fold cross-validation and prints the performance result in exactly the same way as was done for SVM.

Selection of feature set was done using a mixture of TF-IDF and Bag-of-ngrams. The following algorithm was used

```
val = int(ufd[ngramtype] * math.log(TOT_NUM_DOCS/df[ngramtype]));
if val != 0:
    features[ngramtype] =  1
```

NLP First Assignment

Jagat Sastry Pudipeddi (SB ID 108721027)

This is to make sure that those ngrams that occur in most of the documents are not given weightage, while the fact that an ngram occurs is considered more significant than the exact number of occurrence of the ngram (since NB classifier considers feature+value itself as a feature).

Since NLTK NB as well as SVMLight don'tignores any new ngram that it encounters in the test document. Hence some deviation can be observed in the performance when compared to that in the paper.

NLP First Assignment
Jagat Sastry Pudipeddi (SB ID 108721027)

Task 2 (Deceptive review that is difficult for humans to detect)

This is the review that I have written, which deceives most of the humans into believing it is true. But our trained classifiers detect it as being deceptive.

I stayed in Hilton for two days and I was quite satisfied with it. Despite being very inexpensive, the hotel was of good quality. Lake Superior was beautiful and the sunrise over the lake was just spectacular. In addition to many other amenities, the luxurious pools were an added plus. The lush bathrooms and spa center and the luxurious bedroom made my stay worth the $256 that I spent. The staff were very courteous and attentive and the mushroom pizza we were offered on the first day deserves a special mention. They also have this custom of offering exquisite wine to new guests, by which I was very impressed. I travel very often and so far I have been staying in other hotels. Next time I travel to Chicago, I will make sure I stay in this hotel and I highly recommend this hotel to anyone traveling to Chicago. Thank you for a great stay!

Reasons why it will fool humans

1.

Task 3 (Bonus Task)

As specified in the table below, I also tried using POS themselves as features, alongside ngrams. My rationale was that while the ngrams would consider the occurrence of those specific ngrams and the sequence in which the tokens would occur exactly, using POS themselves as feature would give some more information to the classifier, about the frequency and sequence of the POS in a document. Also, if an ngram occurs in the test that doesn't occur in the training set and smoothing is used, it doesn't consider the actual word, instead treating it as *some-unknown word*, unlike it's POS which does have a semantic relationship with the actual word. This is similar to the backoff technique, only that instead of backing off to an (n-1)gram, we backoff to its POS tag.
So while the usage of POS alone would overfit the training set and usage of ngram+ would be useful if the words in the test set are a subset of the words in the training set, POS+NGRAM^{+} would have the benefit of the specificity of the ngram approach while being able to retrieve some information from the new ngrams that occur in the test set.

This feature can be activated by commenting line 120 and uncommenting line 121 in nlp_common.py

```
120     #if CLASSIFIER_TYPE is "POS":
121     if N_IN_NGRAM == 1:
```

As observed in the table below, the performance measures (of those in bold) are higher than the best SVM based approaches that are tabulated in the first page.

Features	Accuracy	Truthful			Deceptive		
POS + UNIGRAMS$_{SVM}$	89.38%	**91.56%**	86.75%	89.09%	87.41%	**92.00%**	89.65%
POS + BIGRAMS$^{+}_{SVM}$	**90.62%**	90.32%	91.00%	90.66%	90.93%	90.25%	90.59%
POS + TRIGRAMS$^{+}_{SVM}$	90.50%	88.76%	92.75%	**90.71%**	92.41%	88.25%	90.28%

Table: Automated classifier performance for the approaches based on nested 5-fold cross-validation experiments. Reported precision, recall and F-score are computed using a micro-average, i.e., from the

NLP First Assignment

Jagat Sastry Pudipeddi (SB ID 108721027)

aggregate true positive, false positive and false negative rates, as suggested by Forman and Scholz