Natural Language Processing, First Assignment
Jagat Sastry Pudipeddi (SB ID 108721027)

Task 1 (Details of my implementations and summary of performances)

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%
$\text{UNIGRAMS}_{\text{SVM}}$	89.75%	**90.77%**	88.50%	89.62%	88.78%	**91.00%**	**89.88%**
$\text{BIGRAMS}^{+}_{\text{SVM}}$	90.00%	88.83%	91.50%	80.15%	91.24%	88.50%	89.85%
$\text{TRIGRAMS}^{+}_{\text{SVM}}$	**90.38%**	88.36%	93.00%	**90.62%**	92.61%	87.75%	90.12%
$\text{UNIGRAMS}_{\text{NB}}$	83.12%	76.55%	**95.50%**	84.98%	**94.02%**	70.75%	80.74%
$\text{BIGRAMS}^{+}_{\text{NB}}$	88.62%	84.56%	94.50%	89.26%	93.77%	82.75%	87.92%
$\text{TRIGRAMS}^{+}_{\text{SVM}}$	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 invoked 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.

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 $\text{UNIGRAM}_{\text{SVM}}$ 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
```

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 ignore any new ngram that it encounters in the test document and there's no smoothing done, some deviation can be observed in the performance when compared to that in the paper.

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 my 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 I was 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 I think it will fool humans

1. It starts off with a moderate tone – *quite satisfied, inexpensive, good quality*. Doesn't seem suspicious at all. No explicit use of superlatives.
2. The reviewer talks in first person, and that increases the credibility, like the paper mentions.
3. It is specific – It talks about Lake Superior, sunrise, pools, spa center, mushroom pizza and welcome wine. Specificity makes the review convincing.

All in all, the reviewer makes it sound convincing enough for a reader to believe him.

However, the classifiers have found it out and these are why I think they did

1. Over use of first person pronouns fools humans, but actually makes the classifier think that it deceptive. The paper discusses the reason why first person pronoun appears more in deceptive reviews, contrary to what our intuition says. Another dimension I could provide to this observation is that the probability of people traveling alone and staying at a hotel is far lower than the probability of them traveling in groups, either with business partners or with family. So the use of first person pronoun will steer the review in deceptive direction.
2. It doesn't explicitly use superlatives or interjections, which causes it to appear less suspicious to humans. However, the hyperbole employed in this review is subtle and are capable of passing human judgment. E.g., spectacular, added plus, lush, luxurious, very impressed, highly impressed, special mention. Moreover, paid or careful reviewers are more likely to use such adjectives to make it look more credible, which will make the classifier aware of such adjectives as deceptive cues.
3. It is specific, but it could be wrongly specific. It could be mentioning specific items that no other truthful reviewers have mentioned or those that deceptive reviewers tend to mention more. For example, there could be no hotel near Lake Superior and the special “mushroom pizza” could be a lie. Specific items that don't occur in the training could result in low smoothed probabilities in either truthful label or deceptive label. If the probability in deceptive label is higher than that in truthful label, it could result in the review being steered towards it being classified deceptive.

All the ngram$_{svm}$ classifiers, detected it as deceptive.

Automation

To test any other deceptive review, insert the review in the file external_review.txt and run

`./external_test.sh $N` ,to train the n-gram SVM classifier using the 800 reviews and predict if the reviews in external_review.txt is deceptive or not. The predictions I obtained with the Ngram SVM classifiers are

1-gram : 1.1941407
2-gram : 0.83160132
3-gram : 0.53464115

One observation from this is that the words themselves were more significant in classifying the review as deceptive, rather than the word sequences.

Task 3 (Possible improvements)

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 == 0:
```

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 + $\text{TRIGRAMS}^{+}_{SVM}$	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 aggregate true positive, false positive and false negative rates, as suggested by Forman and Scholz

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

Accuracy of both POS + BIGRAMS^{+}_{SVM} and POS + $\text{TRIGRAMS}^{+}_{SVM}$ are better than those (my implementations) of BIGRAMS^{+}_{SVM} and $\text{TRIGRAMS}^{+}_{SVM}$ by **0.24% and 0.12% respectively**.

It will be interesting to exactly replicate the performances as given in the paper and then modify the features to use POS + NGRAMS to compare the possible improvement over NGRAM + LIWC or $\text{BIGRAM}^{+.}$