NLP Final Project
Deception Detection

Danu Caus (GUI and Vocabulary Augmentation)
Nambiar Shruti Surendrakumar (Feature Extractors and Testing)
Nana Baah (Research and Data Analysis)
Sebastian Lembcke (Feature Extractors and Testing)

29.01.2019

1. Problem Description

In this project, we aim to analyze how linguistic properties of statements can be utilized to infer if they are truthful or deceptive. We try to find common syntactical patterns in lies and truths, as well as determine which topics people tend to lie about and which terms are used for this purpose. We then apply the knowledge to implement a classification pipeline, which is trained and tested on sentences from a labeled dataset collected by crowdsourcing [2].

2. Dataset

For this task we use the labeled Open Domain Deception Detection Dataset from [2]. It consists of 7168 statements, out of which half are lies and the other 50% are truths. The dataset was collected by means of crowdsourcing from 512 users. Each user stated 7 false and 7 true statements. Since it is open domain, there are few restrictions on the type of sentences collected. Non-commonsensical statements were removed. The dataset also contains metadata, such as age, gender and education level, which we did not use in our experiments, since the purpose was mainly to study the prediction potential of language cues. In a preprocessing step, we extracted all sentences with their corresponding labels, shuffled them and exported them to a training and a test file at a 50/50 split ratio.

3. Software

For our experiments we used the UIMA framework with ClearTK library. We utilized the classification pipeline from the previous exercise on news classification and modified it by adding several feature extractors and functions to compute and visualize different metrics on the new data and topic of deception detection.

For building a better vocabulary than the one present in the dataset (a necessary step for better training results) we've created a separate project package: **AugmentVocabulary** and used for this purpose the **WordNet 3.1** version of the open source princeton dictionary [4] in conjunction with the **MIT JWI API** [5]. This new package basically takes in a file with input words and finds a list of **synonyms** and other related output expressions: **hypernyms** and **hyponyms** as well as different related words belonging to **various parts of speech**.

The application also offers a graphical user interface built with standard **Java Servlet Technology**. The backend module is connected to the **PipelineMain** class and accesses result files to read and return information to the browser using **JSP** and **HTML** pages. Those pages are styled using **CSS**, **Javascript** , as well as the **Bootstrap** and **JQuery** libraries. The files are served/interpreted and executed by an **Apache Tomcat 7** server [6]. As hosting we used a free virtual cluster on the **Heroku** platform [7]. In order to allow a user to test live whether an input statement is truthful or deceptive, an extra class was added in the writer package, namely: **EvaluateUserInput.java**

3.1 Feature Extractors

- First person pronoun extractor: A common honesty feature is the usage of self-references like first-person pronouns. Hence, this feature extractor checks for the presence of such pronouns to extract a boolean feature.

- Negation extractor: highly motivated deceptive people often use more negative statements. Therefore, this feature extractor looks for the presence of such cues as e.g. “don’t”, “won’t”, “wouldn’t” etc.

- Exaggeration extractor: This feature extractor takes words into account that indicate exaggeration, these being e.g. love, hate, never etc. People tend to use exaggeration when lying as a common persuasive technique.

- Lexical extractor: For this extractor we have built a vocabulary from the training set by tokenizing the statements, removing punctuation and stopwords and saving the labeled words to a file. The feature extractor would then count the number of words that are labeled as each category. The assumption is that there are common words used mainly for lying over telling the truth and vice-versa.

- Syntax Features Extractor: This feature extractor makes use of the parsers from the Stanford Core NLP library [9] to obtain syntactic features that are based on part-of-speech tags, dependency types and context-free phrase structures e.g.: A noun-phrase consisting of a noun followed by a specific word may be a predictor of deceptive behavior or to the contrary.

- Readability Scores Extractor: Readability refers to the effort required by users to comprehend the meaning of a specific utterance. This feature extractor incorporates some standard readability scores and syntactic complexity measures which help indicate text understandability. We make use of some of the Readability Metrics APIs provided by Panos Ipeirotis [8].

Below is a high level component diagram of our system. In green are the parts we created ourselves and in yellow are the parts that were inspired by existing components, which however were significantly modified for our purposes. The components in blue are mainly taken from the previous core project of news classification with minor changes to ensure compatibility with the new components.

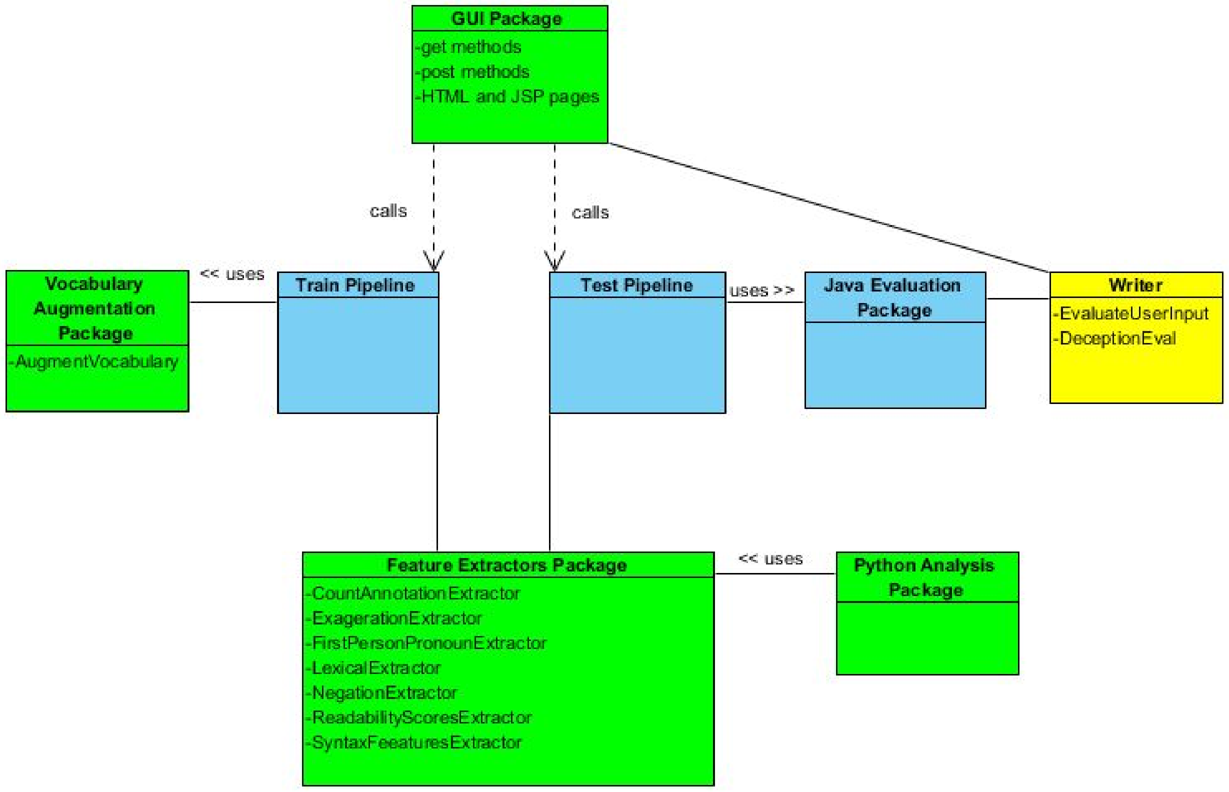


Figure: System Component Diagram

3.2 Software Architecture This NLP application is a data driven software system that follows a model view controller (a.k.a. MVC) structure as seen in the block diagram below

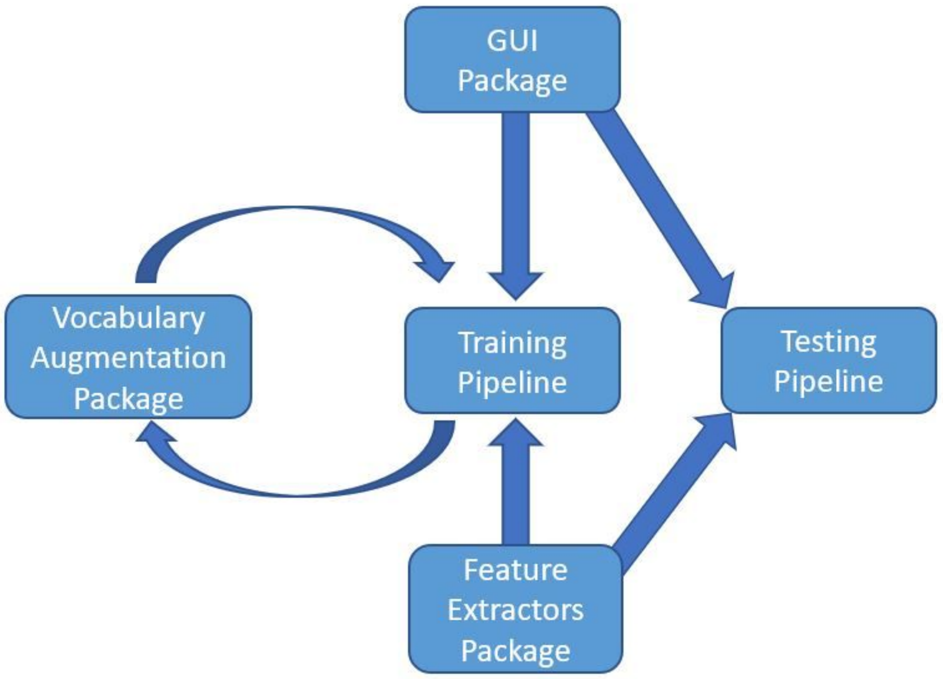


Figure: Full-Stack Application Block Diagram

4. Dataset Analysis

4.1. LIWC

As part of the initial data analysis, we made use of a third-party application called “LIWC”, i.e. Linguistic Inquiry and Word Count. As the author mentions in [3], it reads a given text and counts the percentage of words that reflect different emotions, thinking styles, social concerns, etc. To achieve a valid analysis, the training data set is separately categorized in “truth” and “lie” text files. The LIWC analysis toolkit processes and compares each of these data sets for the most dominant semantic word classes. LIWC also accurately identifies emotion in language use. The positive and negative ratings of emotion words are established via conducted experiments with human ratings as ground truth.

Variables	Lie	Truth
Emotional Tone	50.70	99.00
Dictionary word count (Dic)	66.16	75.40
Function Words (Negation – Negate)	1.59	0.89
Affect	3.79	13.90
Affect (Positive Emotion - Posemo)	2.56	12.80
Affect (Negative Emotion - Negemo)	1.22	1.07
Social	5.81	5.33
Cognitive Processes (Cogproc)	6.09	15.14
Certainty (Certain)	1.77	10.75
Perception Processes (Feel)	0.38	0.53
Biological Processes (Sexual)	0.15	0.08

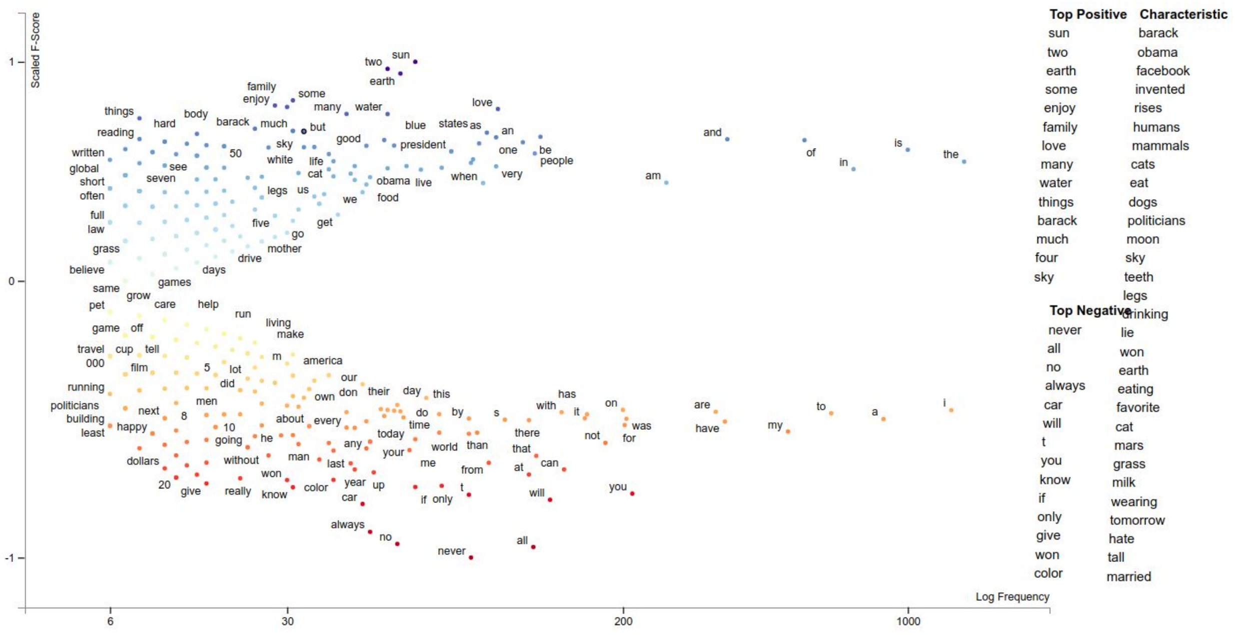
A few LIWC variables are calculated differently from others. Nevertheless, most of the LIWC output variables are percentages of total words within a text. The aim is to distinguish the differences among deceptive and truthful statements. With respect to the emotional tone, the higher the number, the more positive the tone. Values below a threshold of 50 represent a more negative tone.

The results obtained from LIWC analysis highlighted that truth text has a high positive tone of 99%. Intuitively, truthful statements are emotionally positive, whereas lies are negative.

4.2. Visualization

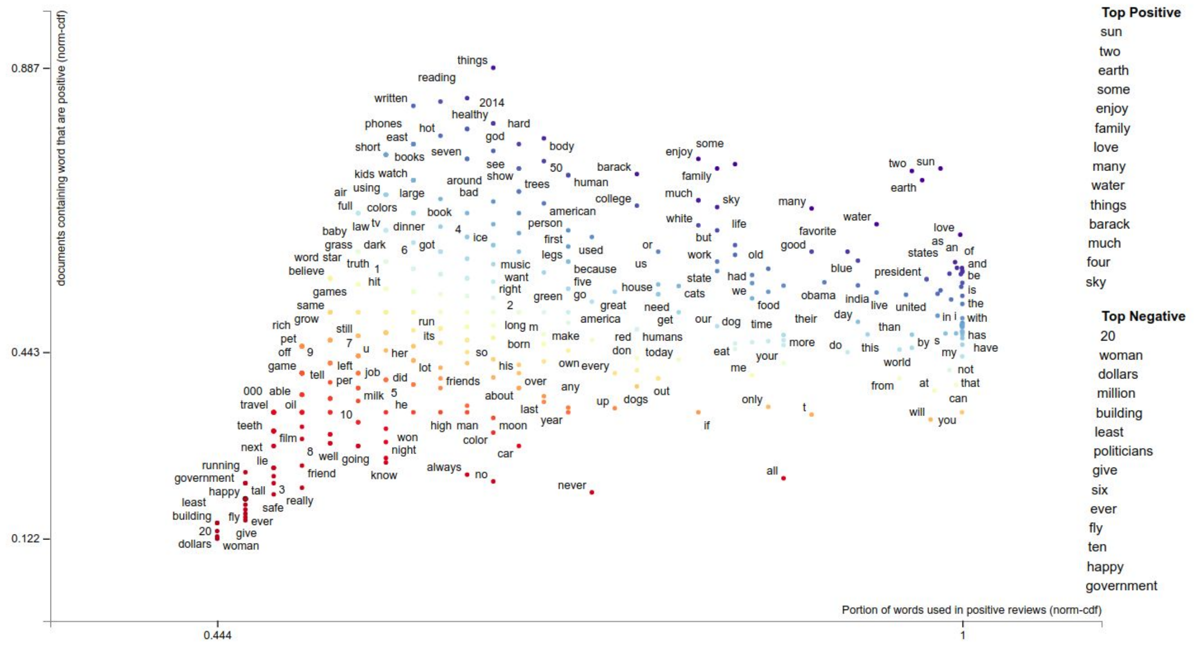
The dataset was analyzed with the Python **Scattertext** tool, that is very popular for visualizing what words and phrases are more characteristic of a category than others.

The figures below show actual word clusters based on our dataset:



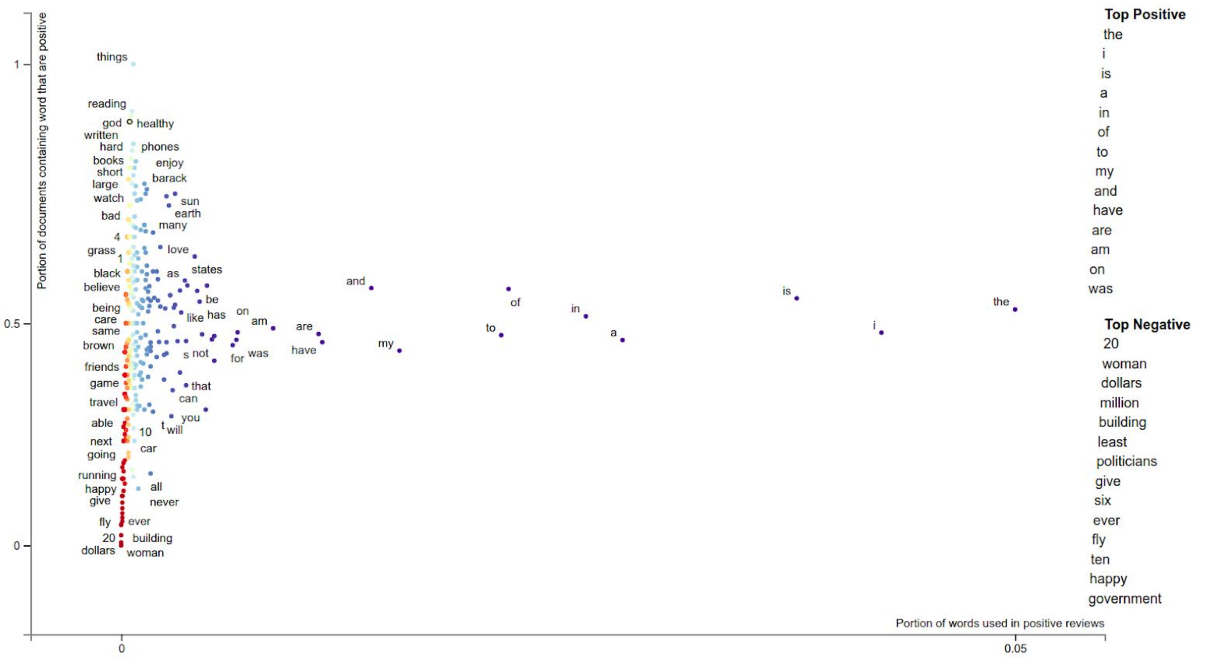
Positive document count: 1,773; word count: 14,329
Negative document count: 1,811; word count: 15,065

Figure: Normalized Frequency Precision



Positive document count: 1,773; word count: 14,329
Negative document count: 1,811; word count: 15,065

Figure: Unnormalized Frequency Precision



Positive document count: 1,773; word count: 14,329
Negative document count: 1,811; word count: 15,065

5. Method and Results

From [1] we decided to incorporate some of the proposed syntactic features and readability features. According to [12] (Tausczik and Pennebaker, 2010) honesty and cognitive complexity are correlated. Since cognitive complexity can be reflected in sentence structure, it could be beneficial to consider syntax based features. For this, we made use of the Stanford Core NLP package, as it offers Java implementations for probabilistic context free grammar (PCFG) parsing. Depending on the properties configured, the parser provides information like part-of-speech tags, CFG structure / grammar representation trees and typed dependency representation (grammatical relations). For obtaining a simpler representation of the dependencies we used **Pikes** (a Knowledge Extraction Suite [10]). The table below shows the kind of information we were able to extract from the parser given an input sentence.

Input	I made a continental breakfast for my roommates today.
Part-of-speech tagged sentence	I/PRP made/VBD a/DT continental/JJ breakfast/NN for/IN my/PRP$ roommates/NNS today/NN ./.
Parsed Tree	(S (NP (PRP I)) (VP (VBD made) (NP (DT a) (JJ continental) (NN breakfast)) (PP (IN for) (NP (PRP$ my) (NNS roommates))) (NP-TMP (NN today))) (. .))
Dependency Information	DepParseInfo{depParents={1=2, 2=0, 3=5, 4=5, 5=2, 6=8, 7=8, 8=2, 9=2, 10=2}, depLabels={1=nsubj, 2=root, 3=det, 4=amod, 5=dobj, 6=case, 7=nmod:poss, 8=nmod:for, 9=nmod:tmod, 10=punct}}

Table: Sample Input-Output when using the Stanford NLP Parser

Now that we had all this data from the parser, we looked into what kind of feature extractors could help in the deception detection domain. The parsed tree feature is implemented with the help of *TreeFeature* class from ClearTK and provides a modest improvement in the F1 score over the baseline. This is consistent with what was reported in [1] and hence this feature is added to the best set of features in all cases.

We also used some psycholinguistic features as suggested by [12] (Tausczik and Pennebaker, 2010) and [13] (Porter and Yuille, 1996). The table below summarizes the features we've tried and the corresponding standalone (using one feature type at a time) F1 scores from the classifier. The standalone features are ranked in descending order according to their scores. Note - the discourse elements dependency includes items like interjections (oh, uh-huh, Welcome), fillers (um, ah), and discourse markers (well, like, actually, but not you know)

Index	**Syntax Feature Type**	**F1 Score**
1	Number of Prepositions	0.56891
2	List dependency present	0.56714
3	Number of Conjunctions	0.56647
4	Discourse elements present	0.56647
5	Number of Adverbs	0.56597
6	Number of Verbs	0.56556
7	Exclusion words present	0.56530
8	Number of Adjectives	0.56529
9	Number of Nouns	0.56384
Combined 1-4	*Top 4 features*	*0.56849*

Table: Syntactic Features and corresponding F1 scores

It was suggested that in order to maintain a fabricated story, people often make use of more cognitively complex constructions. This can be tested with features like the number of conjunctions and presence of exclusion words (eg. but, without, exclude, except, only, just, either) which are often used when differentiating between competing solutions. While the former feature performed well, the latter turned out to be detrimental. The noun and adjective counts extracted from the POS tags data also performed poorly. These counts are roughly supposed to represent simulation attributes often found when descriptions are embellished (as suggested by [11]). These results are likely because of the kind of data we work with - the sentences are long enough to contain conjunctions, but too short for the presence of exclusion words. Also, most of the inputs focus on one core topic, instead of getting into lengthy discussions on different ideas. In the end, we decided to incorporate the **top four performing features from this group** into our best feature set.

The other features, i.e. **exaggeration words**, use of **first person perspective** and the use of **negation** were inspired by [1]. We did however produce our own implementation, which extracts words based on a hard coded dictionary. We obtained the dictionary by analyzing the data manually. The vocabulary for the lexicon based extractor was obtained using a Python script that would tokenize each sentence and create two files, one with all words used in lies and one with all words used in truths. Duplicates and colliding words were removed.

As discussed before, writing deceptive content requires additional cognitive effort and could potentially be more challenging than articulating the truth. To measure this cognitive effort, readability scores and syntactic complexity features can be helpful. To obtain these measures we made use of the readability metrics implementation by Panos Ipeirotis [8]. After testing various configurations of feature sets, we found that the combination of the following indices worked best for our feature set, giving a slightly higher F1 score over the baseline:

- ➔ **Flesch Kincaid** (a measure of readability that indicates the comprehension difficulty when reading a passage of contemporary academic English. Lower scores imply the material is easier to read.)
- ➔ **Smog grade** (a measure of readability that estimates the years of education needed to understand a piece of writing)
- ➔ **Number of characters** and **number of syllables** in the sentence

Our final set of selected features extractors are summarized in the table below -

Best Feature Extractor Set
Lexical Extractor
Syntax Features Extractor (parsed tree, number of prepositions, presence of lists of items, number of conjunctions, presence of discourse elements)
Readability Scores Extractor (Flesch Kincaid score, Smog grade, number of characters and number of syllables)
First person pronoun extractor
Negation Extractor
Exaggeration Extractor

On the whole, the above mentioned feature extractors did not improve the performance significantly. In fact, the difference varied only at around +/- 0.03 (as can be seen in the table below), which could also be an effect of randomisation.

Scenario/ Metric	**Baseline**	**Best Feature Set without vocabulary augmentation**	**Best Feature Set with vocabulary augmentation**
Precision	0.5617701290719115	0.5606936416184971	0.5633314054366686
Recall	0.544047619047619	0.5773809523809523	0.5797619047619048
F1 score	0.552766858179619	0.5689149560117301	0.5714285714285715
Accuracy	0.5611275964391691	0.5637982195845698	0.5664688427299703

6. Conclusion

In general we can say that, in the open domain context, the language cues are not very successful in identifying whether a person is truthful or not. In order to have more accurate results, the language features have to be accompanied by various socio-economical traits like age, gender, level of education etc. This claim is postulated and supported by [1].

We can also confirm that negation is not a very strong feature, contrary to what we expected initially. This could stem from the fact that a negation can have a very ambiguous scope, as well as because it is used randomly in deceptive/truthful behaviours.

Moreover, we can say, that other patterns, such as exaggeration or the use of first person perspective, are not reliable features as they are also widely used in both: truth and lie scenarios.

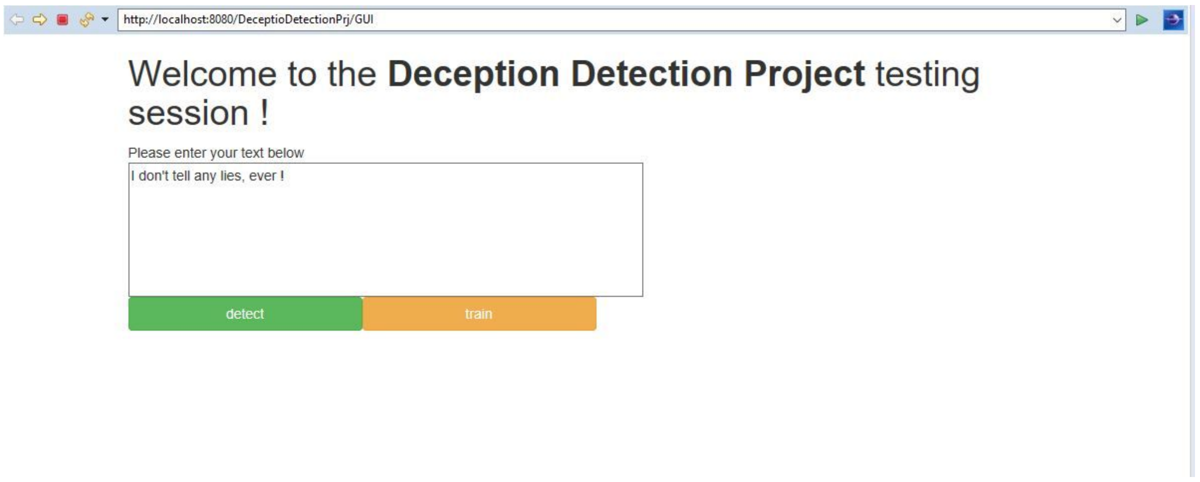
The lexicon based feature extractor also did not improve the performance of the system significantly. It is however, a promising approach, as it could extract common topics associated to lies and truths and seems to also work especially well in conjunction with the “Syntax Features Extractor” and the “Readability Scores Extractor”.

The selected subset of Syntactic and Readability features did manage to give a higher marginal improvement in the results, but these need further investigation. Taking traits like age into account [14] (Yancheva and Rudzicz, 2013) can be beneficial for the readability metrics, as they are often correlated.

7. Graphical User Interface

The GUI offers the possibility to classify on the fly various statements entered in a textarea by clicking the **detect** button. Alternatively, by clicking the **train** button, it also allows the user to see the 4 most important metrics in a bar chart, calculated based on the test dataset.

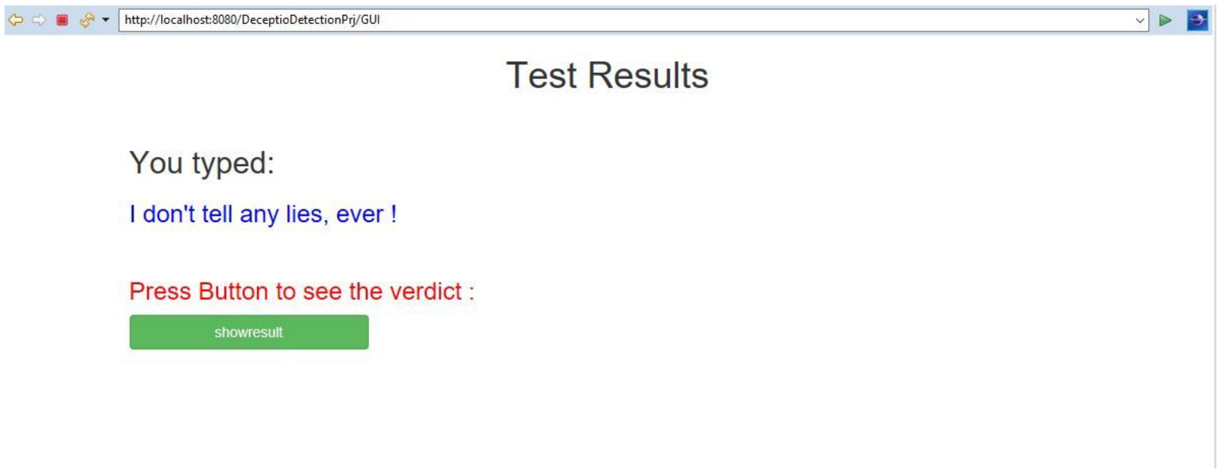
It can be checked live at: https://deceptiondetection.herokuapp.com/GUI or visualized via the image snippets below (taken from the built-in Eclipse browser)


http://localhost:8080/DeceptioDetectionPrj/GUI

Welcome to the **Deception Detection Project** testing session !

Please enter your text below

I don't tell any lies, ever !

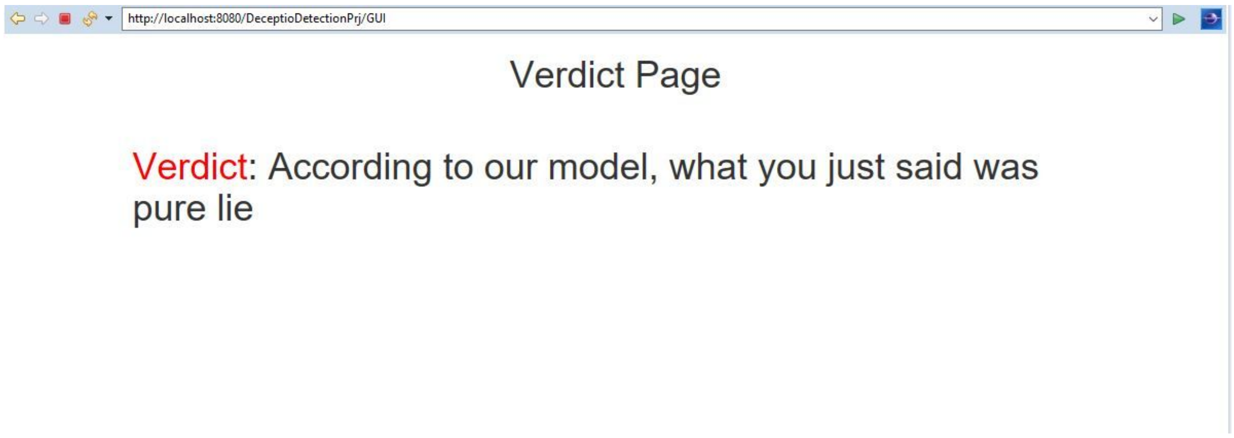

http://localhost:8080/DeceptioDetectionPrj/GUI

Test Results

You typed:

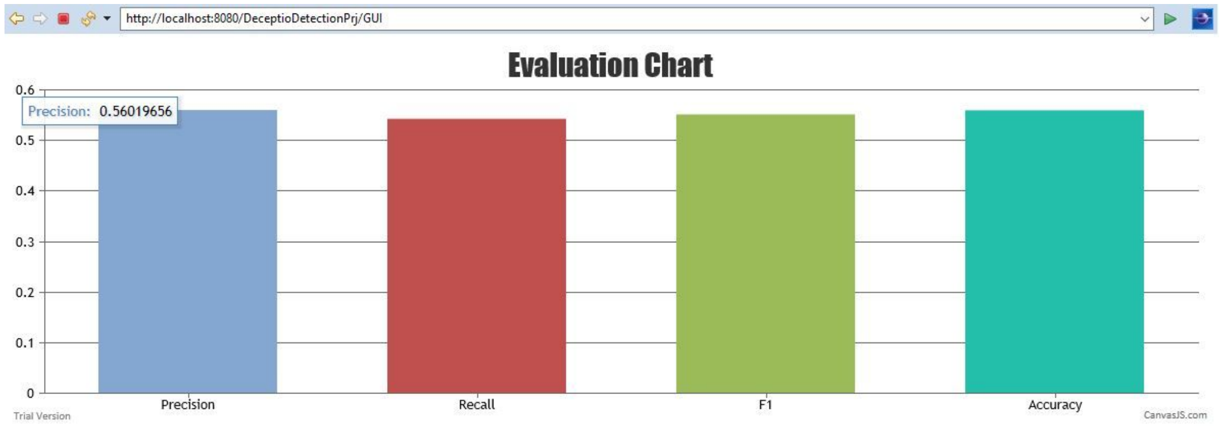
I don't tell any lies, ever !

Press Button to see the verdict :


http://localhost:8080/DeceptioDetectionPrj/GUI

Verdict Page

Verdict: According to our model, what you just said was pure lie


http://localhost:8080/DeceptioDetectionPrj/GUI

8. Future Work

Since this is essentially a machine learning problem, a lot of quality data is always a good and useful thing to have. This is true especially since in our case, we had less than 8000 sentences, some of which having missing labels unfortunately.

Our vocabulary augmentation can be improved by using other online semantic databases like **word2vec** for example, as well as extracting extra linguistic features from them like: **antonymy** or **homonymy** relationships, **pejorative** and **metaphorical** meanings, **figures of speech** etc.

In order to improve results, it is mandatory also to use other **non-linguistic** features as suggested by [1], based on existing statistical evidence that people are strongly influenced by their socio-economic status as to whether they will lie or tell the truth.

References

[1] Veronica Perez-Rosas and Rada Mihalcea, Experiments in Open Domain Deception Detection, in Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP 2015), Lisbon, Portugal, September 2015, http://web.eecs.umich.edu/~mihalcea/papers/perezrosas.emnlp15.pdf

[2] Open Domain Deception Detection Dataset
https://web.eecs.umich.edu/~mihalcea/downloads.html#OpenDeception

[3] Linguistic Inquiry and Word Count http://liwc.wpengine.com/

[4] Wordnet Database Files (A lexical database of English which records semantic relationships between words)
https://wordnet.princeton.edu/

[5] Java API for accessing wordnet https://projects.csail.mit.edu/jwi/

[6] Apache Tomcat 7 Java Server http://tomcat.apache.org/tomcat-7.0-doc/

[7] Heroku Cloud Application Platform https://www.heroku.com/

[8] Panos Ipeirotis. REST API: Retrieving Text Metrics.
https://github.com/ipeirotis/ReadabilityMetrics/wiki/REST-API%3A-Retrieving-Text-Metrics

[9] Stanford Core NLP Parser https://nlp.stanford.edu/software/lex-parser.shtml

[10] Pikes - Knowledge Extraction Suite. https://github.com/dkmfbk/pikes

[11] Girlea et al. Psycholinguistic Features for Deceptive Role Detection in Werewolf.
http://www.aclweb.org/anthology/N16-1047

[12] Yla R Tausczik and James W Pennebaker. 2010. The psychological meaning of words: Liwc and computerized text analysis methods. Journal of language and social psychology, 29(1):24–54.

[13] Stephen Porter and John C. Yuille. 1996. The language of deceit: An investigation of the verbal clues to deception in the interrogation context. Law and Human Behavior, 20(4):443–458.

[14] M. Yancheva and F. Rudzicz. 2013. Automatic detection of deception in child-produced speech using syntactic complexity features. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 944–953, Sofia, Bulgaria, August. Association for Computational Linguistics.