Fake News Detection

Hyunwoo Sohn, Kyungjin Park, Jaymin Desai, Shwetha Kalyanaraman
North Carolina State University
`hsohn3, kpark8, jddesai2, skalyan` @ ncsu.edu

Abstract

We present an analytic study on the fake news detection with various machine learning techniques. We examined LIAR dataset[1] for political fact-checking, and used Bag-of-Word (BoW), Term Frequency - Inverse Document Frequency (TF-IDF), and a word embedding method, Word2Vec, for text processing. In order to figure out the most suitable model for detecting fakeness, we compared the performance of three standard classifiers and two deep learning models. We further investigated the sentiment feature extraction to evaluate the effectiveness of the factor to detection. The result shows that the deep learning models perform higher accuracy, while the standard machine learning classifiers show higher f-measure, and the sentient feature addition does not significantly improve the performance of the models. All the experimental codes are hosted on Github [2].

1 Background

Fake news is a digital generation problem which influences people to take quick decisions without understanding the truth behind it (1). The emergence of social media makes it easier to read, share, and discuss the news articles compare to the traditional newspapers and magazines, which enables the fake news to be easily distributed among the people. This extensive spread of the fake news creates a serious problem in the society, often breaks the authenticity of the news ecosystem and intentionally makes readers accept the false-beliefs (2).

In order to mitigate the effects caused by the proliferation of fake news online, it is crucial to detect them before it reaches many. Many researchers have studied deceptive detection (3), but it is still challenging to accurately predict fakeness due to the underlying tricks that make the news look more concrete and trustworthy. Followings are the previous approaches in the fake news detection research.

- **Linguistic Approaches**: Many researches have been based on the linguistic feature extraction for classifying the news as fake or not. Satire hides its meaning in irony and double-meanings which could help us in the detection (4). Other features such as humor, absurdity and grammar, are also proven to be a good factor to relate the sentiment behind the sentence (5). Punctuation plays a major role in satirical statements. These features are truly qualifying factors for the prediction, but it takes human labors to analyze each of the linguistic characteristics.
- **Machine Learning Approaches**: The proliferation of machine learning helps on further research in deceptive detection using various methods. Mikolov and colleagues proposed two models for computing continuous vector representations of words from very large data sets (6). The models measure the similarity between words which makes it easier to extract the underline meanings of the sentences without human effort.

[1]`https://www.cs.ucsb.edu/~william/data/liar_dataset.zip`
[2]`https://github.com/jaymindesai/fake-news-predictor`

2 Related Work

2.1 Fake News Detection using Hybrid Neural Network

Our model is inspired by the hybrid deep learning model proposed by (7), which was designed to integrate meta-data with text to enhance the performance of fake news detection. This model applied a convolutional neural network(CNN) for the text embeddings, and a combination of CNN and a bi-directional LSTM(Long Short-Term Memory) for the speaker related meta-data embeddings. The text representations and the meta-data representation from both channels were concatenated and fed to fully connected layer to generate the final prediction. The inputs for the model were both word vectors generated by Google pretrained 300-dimensional Word2Vec model (8). The (7) used the state-of-the-art models and experimented with several combinations of features. However, it has not considered any other features which could be obtained from the text.

2.2 Sentiment Analysis

Sentiment analysis is an active field of natural language processing that analyzes people's opinions, sentiments, evaluations, attitudes, and emotions via the computational treatment of subjectivity in text (9). It mostly depends on sentiment lexicon which are labeled according to their semantic orientation as either positive or negative. The valence aware dictionary for sentiment reasoning(VADER) model (9) examined all features of existing well-established sentiment lexicons (Linguistic Inquiry and Word Count - LIWC, Affective Norms for English Words - ANEW, and General Inquirer - GI). By supplementing the lexicons commonly used in social media, it acquired 7.5K lexical features as a gold-standard lexicons. It evaluated the effectiveness of the model using 11 state-of-practice benchmarks including machine learning oriented techniques, but not any deep learning models have been applied.

3 Research Questions

We set two research questions toward better fake news detection.

- **RQ1**: How well can machine learning algorithm detect fake news, and which model would perform better in classifying fakeness?
- **RQ2**: What kinds of representation could we obtain more from the text itself, are we able to improve our model by using those additional features?

To answer the first question, we compare the performances among three standard classifiers and two machine learning models. Once we figure out the best model, we add features to examine their effectiveness.

4 Method

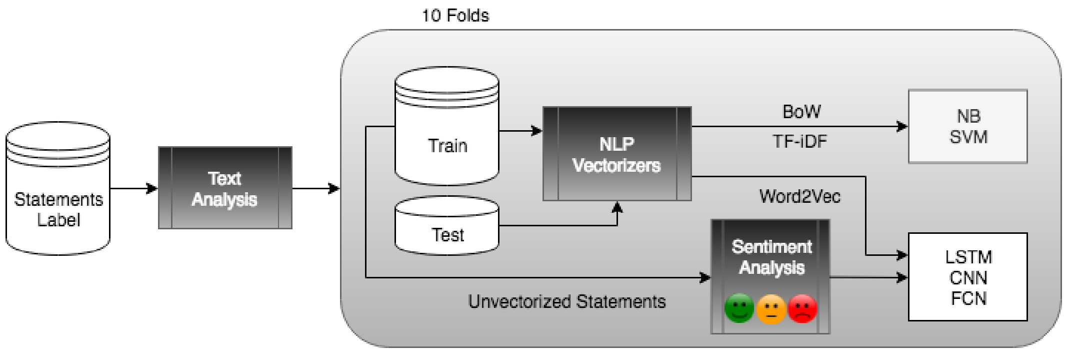

Figure 1: Fake News Detection Framework

In this work, we focused on applying different NLP and Machine Learning techniques to the ***statement*** feature in order to predict the truthfulness as shown in Figure 1.

4.1 Text Analysis

- **Preprocessing**: We designed a tokenizer that uses Natural Language Toolkit (NLTK) and performs the following sequence of events:
  - Break the statements paragraph into separate sentences.
  - Preprocess the token by converting all the letters to lower case, strip whitespaces and other punctuations, remove other unnecessary symbols like underscore or asterisk.
  - Ignore the token if its a stopword which refers the most common words in a language [wiki] (e.g. the, a) or a digit.
- **Weighing Text Features**: We used two different weighing schemes to transform the *statements* into Stetement-by-Terms matrix.
  - **Term Frequency - Inverse Document Frequency**: We used TfidfVectorizer from SciKit-Learn to obtain a matrix where each row contains the weights of the terms occurring in that particular statement. This scheme gives higher weights to words that appear least often.
  $$tf-idf_{t,d} = tf_{t,d} \cdot \log \frac{|D|}{1+df_t} \tag{1}$$
  - **Bag of Words**: We used CountVectorizer from SciKit-Learn to obtain a matrix where each row contains the no. of occurrences for a term in that particular statement.

4.2 Natural Language Processing - Word2Vec

Word2Vec is a neural network model that represent words in a continuous vector space where semantically similar words are mapped to nearby points. By predicting the context words given the center word or vice versa, the model can obtain a meaningful word embeddings. The equation (2) shows how we calculate the probability of "output" words given the center word, where o is context words that are around the center word, and c is the center word. vector u and v indicate the vector representations for each context and center word(8).

$$P(o \mid c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{V} exp(u_w^T v_c)} \tag{2}$$

The loss function in (3) simply use the probability shown in (2) and the model tries to minimize it to have an optimal word embedding, where T is the number of words in the text, m is the window size, and t is the position where the center word lies.

$$J(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log P(W_t + j \mid W_t; \theta) \tag{3}$$

We employed two versions of word embeddings, one trained on our own dataset the LIAR, and the other pre-trained by Google[3] with a part of Google News dataset(about 100 billion words) to check if domain specific semantics help fake news prediction.

4.3 Learning Classifiers

This section discusses how we used machine learning techniques to classify news into six and two different classes respectively. We used Naive Bayes(NB) and Support Vector Machine(SVM) as our preliminary classifiers followed by Deep Learning and other NLP techniques. The choice of using NB and SVM compared to other techniques is not random as they have been successfully used in many previous works (10) (11) to predict defects in Software Engineering datasets and classify bug reports - text classification in general. The two deep learning models CNN and LSTM have also obtained the state-of-the-art results in sentence classification: (12) (13).

- **Naive Bayes**: We tried eight different permutations using Gaussian and Multinomial Naive Bayes with TF-IDF and BoW vectorized statements with six class labels and two class labels.
$$P(y \mid x_1, \ldots, x_n) = \frac{P(y)\prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)} \tag{4}$$

[3]https://code.google.com/archive/p/word2vec/

- **Support Vector Machine**: We tried the same permutations for SVM as we did with Naive Bayes using *rbf* kernel for 6-class model as the baseline with the understanding that as our input to the model operates in higher dimensions (TF-IDF/BoW matrix), *rbf* kernel is ought to perform better compared to *linear* kernel. We used SciKit-Learn SVC which automatically *scales* the value of gamma from the features and used default penalty *C* = 1.

  $$K(x, x') = exp(-\gamma \|x - x'\|^2) \tag{5}$$

  However, in order to extract other evaluation metrics and additional experiment for 2-class models, we computed the *linear* kernel for the sake of the processing time, since the results of rbf and linear had only slight differences .

  $$K(x, x') = x \cdot x \tag{6}$$

- **CNN**: This model has a basis on the well-known CNN architecture(12) for sentence classification. Initially we convert our statement to word vectors and fed them to convolutional layer. Let x_i be the *k*-dimensional word vector of *i*-th word in a sentence. A sentence of length n is represented as a *n*k* vector, which is generated by concatenating word vectors x_i to x_n. Let $x_{i:i+j}$ refer to concatenation of words x_i to x_{i+j}. A convolution operation involving a filter $\mathbf{w} \in \mathbb{R}^{hk}$ is applied to a window of h words to produce a new feature. For example, a feature c_i is generated from a window of words $x_{i:i+h-1}$ by (7), where b is a bias and f is non-linear function. The filter is applied to each possible window of words in the sentence to produce a feature map, $c = [c_1, c_2, ...c_{n-h+1}]$. Then we apply a max pooling operation to capture the most important feature for each feature map. The model uses multiple filters, with varying window sizes, to obtain multiple features, and these features are passed to a fully connected layer to predict the label.

  $$c_i = f(\mathbf{x}_{i:i+h-1} + b) \tag{7}$$

- **LSTM**: We built our model using a variant of RNN, LSTM. Compared to RNN, LSTM employs three additional gates in hidden layer that help the model avoid the long-term dependency problem by controlling which information to remove or add. The forget gate (8) decide what to drop, the input gate (9) decides which values to update, and the output gate (10) tells the model what to use as an output(14). Since we tried to get benefits from both semantic and syntactic sides of the statements, we used word embeddings that contains semantics of words as an input for LSTM, which can learn dependencies between words.

  $$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \tag{8}$$
  $$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \tag{9}$$
  $$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \tag{10}$$

  Other than using LSTM solely, we combined a fully connected network to integrate sentiment extracted from the statements since sentiment is proven to be a qualifying factor in detecting deception (15). We collected sentiment using VADER sentiment, a lexicon and rule-based model (9). Moreover, we assumed that examining various aspects of the statement using both deep learning model and classic rule-based model would help prediction.

5 Experiment

5.1 Dataset

In this research, we use LIAR dataset (7) which contains 12,791 human labeled short statements from POLITIFACT.COM for the fake news detection. Each statement that was made by a politician in a public event is evaluated by Politifact.com editors for its truthfulness. In order to ensure the validity of the human-labeled data, the researchers randomly sampled and measured the agreement (further discussed in Section 7). The dataset has 12 features including the statement, speaker's name, speaker's job title, and etc., and one output label. (See (7) for the detailed description of the features) Each statement is classified into six labels - *true, mostly-true, half-true, barely-true, false, pants-fire*.

The original dataset is divided into the train, validation, and test data with 10240, 1284, 1267 respectively, but we combined all three, then performed 10 fold Cross Validation. The distribution of the labels is pretty well balanced: Other than 'pants-fire' label (8.19%), all labels are ranges from 16.05% to 20.54%. Furthermore, we repeat all the experiments on a new dataset obtained by combining the six class labels into two (true, mostly-true, half-true as ***true*** and remaining as ***false*** where 'true' implies truthfulness and 'false' implies deception) to see how well the model detect the statement's deception. The distribution of 'true' and 'false' labels is 55.77%, and 44.23%, respectively.

5.2 Experimental Settings

We conducted experiments on the five different models: Gaussian NB, Multinomial NB, SVM, LSTM, and CNN. For NBs and SVM, we used BoW and TF-IDF as their input. Two versions of Word2Vec, custom and pretrained, were fed to the deep learning models. For SVM, we used linear kernel with default penalty C = 1. For LSTM, we used dropout rate of 0.5, 10 epoches, Softmax activation, and Adam optimizer. For CNN, we used filter size of 3, 4 and 5 and a dropout rate of 0.5. We used 3 convolutional networks and applied max pooling on them followed by a softmax activation function. We used window size of 20, 150 dimensions to train the word embedding for LIAR dataset. We used fully connected network(FCN) for sentiment feature extraction with 1 hidden layer, 6 hidden nodes, and sigmoid activation. Each model is trained and tested with 10-fold cross validation.

5.3 Evaluation Metrics

We measure the predictive capability of our models using four different metrics (Recall, Accuracy, Precision and F-measure) but use only Accuracy and F-measure to evaluate our models. Detecting maximum number of fake news is our primary goal, and just Recall would serve the purpose but if we optimize our models for Recall, that will increase false alarms (True news being classified as Fake). We wanted to limit the number of false alarms and as a trade-off we used F-measure to evaluate our models along with Accuracy.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$F - Measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{12}$$

6 Result

We first compared the performance of the standard classifiers and the deep learning models. Table 1 and 2 show the results of classifiers with 6-class and 2-class labels, respectively. For both the cases, deep learning models outperform standard classifiers in accuracy (CustomW2V+LSTM - 26.13%, and (pre-trained)W2V+LSTM - 62.68%), while the standard classifiers outperform the other in terms of F-measure. This result does not fully agree with our hypothesis that the deep learning models would outperform the standard classifiers, but there are many rooms for tuning hyper parameters in the deep learning models while this is the best result we could get, especially with the Naive Bayes models. Thus, there still be many chances we could meet our hypothesis. Furthermore, we combined the sentiment feature to the deep learning models. Table 3 and 4 shows the result of this experiment. Due to the time limit, we only added sentiment feature to the LSTM models since it outperforms CNN in statement-only models. While we got slightly better performance with sentiment feature for 6-class labels, we got the opposite result with 2-class labels. One possible reason is over-fitting and the other is that LSTM was likely to obtain enough representations regarding the fakeness from statement embeddings. As a result, Sentiment feature didn't help and this could be just another form of the statements.

7 Threats to Validity

- **Threats to Construct Validity**: The LIAR dataset we used in our experiments has six different class labels (as discussed in Section 5.1) that are manually labeled by Politifact editors and journalists. This means that the manually labeled truth set that we used to evaluate our models might affect the overall outcome of our experiment as we had to

Table 1: Standard Classifiers vs Deep Learning Models (6 Class)

	6 Class			
	Recall	Precision	Accuracy	F-measure
BoW + Gaussian NB	21.49	22.47	18.97	21.97
TF-IDF + Gaussian NB	21.46	21.95	19.28	21.70
BoW + Multinomial NB	23.33	26.82	25.55	24.96
TF-IDF + Multinomial NB	21.59	38.00	25.17	**27.54**
BoW + SVM	24.87	25.25	25.86	25.06
TF-IDF + SVM	25.29	26.24	26.17	25.76
CustomW2V + LSTM	23.14	27.03	**26.13**	24.93
(pre-trained) W2V + LSTM	24.84	26.38	26.05	25.59
(pre-trained) W2V + CNN	16.90	15.07	21.10	15.93

Table 2: Standard Classifiers vs Deep Learning Models (2 Class)

	2 Class			
	Recall	Precision	Accuracy	F-measure
BoW + Gaussian NB	82.48	46.65	50.63	**59.59**
TF-IDF + Gaussian NB	78.94	47.05	51.56	58.96
BoW + Multinomial NB	51.68	58.82	62.52	55.02
TF-IDF + Multinomial NB	34.51	63.64	61.89	44.75
BoW + SVM	54.69	53.92	59.31	54.30
TF-IDF + SVM	52.38	55.80	60.48	54.04
CustomW2V + LSTM	59.36	58.19	61.89	58.77
(pre-trained) W2V + LSTM	56.89	62.22	**62.68**	59.44
(pre-trained) W2V + CNN	43.58	57.93	59.11	49.74

Table 3: Statement vs Statement + Sentiment feature (6 Class)

	6 Class			
	Recall	Precision	Accuracy	F-measure
CustomW2V + LSTM	23.14	27.03	**26.13**	24.93
CustomW2V + LSTM + Sentiment	24.11	26.78	26.05	**25.37**
(pre-trained) W2V + LSTM	24.84	26.38	26.05	25.59
(pre-trained) W2V + LSTM + Sentiment	25.63	27.32	**26.68**	**26.45**

Table 4: Statement vs Statement + Sentiment feature (2 Class)

	2 Class			
	Recall	Precision	Accuracy	F-measure
CustomW2V + LSTM	57.60	59.61	**62.44**	**58.59**
CustomW2V + LSTM + Sentiment	52.57	59.22	62.36	55.69
(pre-trained) W2V + LSTM	60.53	57.20	61.27	**58.82**
(pre-trained) W2V + LSTM + Sentiment	53.36	58.36	**61.50**	55.74

indirectly rely upon error-prone human judgment. This can't be avoided as there is a level of subjectivity involved in deciding if a *statement* falls under a specific category. However, to alleviate this, in each case, the labeler provided a lengthy analysis report to ground each judgment, and the links to all supporting documents were also provided. A second verification run from the LIAR team was then conducted with a randomly sampled subset of

200 instances to check if the analysis reports were agreeable. The agreement rate measured by Cohen's Kappa was 0.82, significantly beyond the acceptance mark.(7)

- **Threats to Internal / External Validity**: In our experiment, we first compared standard Machine Learning classifiers (Naive Bayes and Support Vector Machine) with Deep Learning models (CNN and LSTM) and decided to further optimize LSTM. We selected LSTM as it was tested to give the best results in the referenced work and hence used the referenced work as a tie-breaker. This might pose a threat to our work as we rely upon other research which may happen to be flawed.

  To limit the scope of our experiment, we focused solely on the *statement*. We tried to improve our model by adding features extracted by performing Sentiment Analysis on the *statement* but neglect various explicit features in the existing database. This may concern the generalization of our findings.

8 Conclusion

Fake news has the potential to create outrageous circumstances in no time and the gravity of this issue is increasing, with each passing day. In this experiment we exploited standard machine learning classifiers and deep learning models to address the issue of identifying fake, deceptive news. We analyzed the performance of four different models on the LIAR dataset and optimized LSTM by adding Sentiment details to the existing model. We transformed the data using NLP heuristics, classified the news as Fake/Truth and measure the performance of the models using Accuracy and F-measure. We employed several different approaches before finalizing on LSTM and experimented with different Word Embedding methods. The novelty of our approach lies in providing sentiment details as an input to LSTM along with vectorized statements and critique the outcome. We also tried to address other issues of reducing the cost of computation-heavy machine learning models by using text clustering but couldn't finish the experiment due to time constraints.

9 Future Work

- **Partial Experiment – Unsupervised Clustering**: Clustering can greatly reduce the training times and possibly the performance metrics as well. The idea was to cluster the data and classify within the cluster bounds. We clustered the dataset with other features like *speaker, party, credibility counts* using Unsupervised K-Prototype Clustering as dataset has mixed features. K-Prototype uses K-Means for continuous numeric features while K-Modes for categorical features (16). But, clustering did not prove useful and we speculate the following to be the primary reasons:

  - The distribution of data points across the feature space must be such that there is an excessive amount of overlap among the fake and truth data points which makes it difficult for K-Prototype to determine coherent cluster boundaries. The fake and truth instances if visualized, would tend to occur in random uneven shapes other than the spherical blobs that K-Prototype tends to work best with.
  - We used Elbow Arc method to determine the number of clusters which sometimes was a uniform curve that gave an ambiguous elbow arc.

  We believe a better alternative would be to use DBScan (Density Based Spatial Clustering of Applications with Noise) (17) which is tested to work much more effectively when the distribution of data points across multiple class labels has a lot of overlap and exists in uneven shapes. A more robust method to select the initial number of clusters like the Average Silhouette method or the Gap Statistic (18) method could also give better results.

- **Hyperparameter Optimization**: While we try to optimize the performance by adding extra features like Sentiment scores, the model we used to report our final metrics is semi-tuned due to the time constraints. As an extension to our experiments, the models (Standard and Deep Learning) that we used can be tuned and optimized for specific goals. The vectorized dataset in every fold has over 11,000 features and tuning the hyperparameters using a complete search approach like Grid Search would be extremely heavy on computation and would take a very long time even on a high performance machine. We suggest using a smarter approach like Differential Evolution that performs stochastic jumps across the hyperparameter space to give results at least as good as Grid Search and is much faster comparatively.

- **Feature Selection using ML Classifiers**: We reduced the number of features by considering only *statement* but the vectorized statements has over 11,000 features and using different feature selection techniques like Principal Component Analysis, Co-relation based Feature Selection and ML Wrappers would definitely help our classifiers to converge in lesser time and improve their predictive power.
- **NLP techniques**: We use Word2Vec for our analysis but there are other word embedding techniques that are found to be equally effective for text classification like GloVe.

References

[1] M. Gahirwal, "Fake news detection," in *IJARIIT, 1 Jan. 1970.*

[2] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, "Fake news detection on social media: A data mining perspective," in *ACM SIGKDD Explorations Newsletter, 2017*.

[3] V. L. Rubin, "Deception detection for news: Three types of fakes,"

[4] H. Rashkin, E. Choi, J. Y. Jang, S. Volkova, and Y. Choi, "Truth of varying shades: Analyzing language in fake news and political fact-checking," in *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

[5] V. Rubin, N. Conroy, Y. Chen, and S. Cornwell, "Fake news or truth? using satirical cues to detect potentially misleadingnews.," in *Proceedings of the Second Workshop on Computational Approaches to Deception Detection*, 2016.

[6] T. Mikolov, "Efficient estimation of word representations in vector space,"

[7] W. Wang, ""liar, liar pants on fire": A new benchmark dataset for fake news detection.," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics.*

[8] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality.," in *Proc. Advances in Neural Information Processing Systems 26 3111–3119 (2013).*

[9] C. Hutto and E. Gilbert, "Vader: a parsimonious rule-based model for sentiment analysis of social media text.," 2014.

[10] E. Ceylan, F. Kutlubay, and A. Bener, "Evaluating static analysis defect warnings on production software," in *in Software Engineering and Advanced Applications (SEAA)*, 2006.

[11] G. Antoniol, K. Ayari, M. Di Penta, F. Khomh, and Y. Guhneuc, "Is it a bug or an enhancement?: a text-based approach to classify change requests," in *CASCON*, 2008.

[12] Y. Kim, "Convolutional neural networks for sentence classification.," in *EMNLP, 2014*.

[13] I. Sutskever, O. Vinyals, and L. Q. V., "Convolutional neural networks for sentence classification.," in *Proc. Advances in Neural Information Processing Systems 27 3104–3112 (2014).*

[14] C. Colah, "Understanding lstm networks : http://colah.github.io/posts/2015-08-understanding-lstms/,"

[15] B. Liu, "Sentiment analysis and opinion mining.," in *San Rafael, CA: Morgan & Claypool, 2012*.

[16] Z. Huang, "Extensions to the k-modes algorithm for clustering large data sets with categorical values," in *Data Mining and Knowledge Discovery 2(3), 1998, pp. 283-304*.

[17] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," 1996.

[18] R. Tibshirani, G. Walther, and T. Hastie, "Estimating the number of clusters in a data set via the gap statistic," 2002.