FII_UAIC at SemEval-2022 Task 6: iSarcasmEval - Intended Sarcasm Detection in English and Arabic

Tudor Manoleasa
Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Romania
tudor.manoleasa@info.uaic.ro

Iustin Sandu
Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Romania
iustin.sandu@info.uaic.ro

Daniela Gifu
Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Romania
Institute of Computer Science, Romanian Academy - Iasi Branch
daniela.gifu@info.uaic.ro

Diana Trandabat
Faculty of Computer Science, Alexandru Ioan Cuza University of Iasi, Romania
diana.trandabat@info.uaic.ro

Abstract

The "iSarcasmEval - Intended Sarcasm Detection in English and Arabic" task at the SemEval 2022 competition focuses on detecting and rating the distinction between intended and perceived sarcasm in the context of textual sarcasm detection, as well as the level of irony contained in these texts. In the context of SemEval, we present a binary classification method which classifies the text as sarcastic or non-sarcastic (task A, for English) based on five classical machine learning approaches by trying to train the models based on this dataset solely (i.e., no other datasets have been used). This process indicates low performance compared to previously studied datasets, which indicates that the previous ones might be biased.

1 Introduction

One of the most challenging tasks facing natural language processing (NLP) is the automatic figurative language detection (Gifu Daniela, Samson Mihai, 2021), (C. Van Hee, E. Lefever, and V. Hoste, 2018), such as humor, sarcasm or irony. In general, this way of expressing takes advantage of linguistic elements in order to project complex and explainable meanings. In this paper, we investigate the binary classification models for figurative language, as is sarcasm (Dan Alexandru and Daniela Gîfu, 2020). In general, it is omnipresent on the public space, being disruptive of computational systems that harness this data to perform tasks such as opinion mining and sentiment analysis in elections (Gîfu, Daniela, 2010). The political actors themselves introduce a specific language based on sarcasm or irony, making their message analysis very challenging (Reyes, Antonio and Rosso, Paolo and Buscaldi, Davide, 2012). In order to identify the figurative meaning of a specific message, it is required to encode each sentence separately. The research question guiding this paper is what are the most efficient sarcasm detection algorithms? We propose an approach based on three classical machine learning (ML) approaches by trying to train the models based on this dataset solely (i.e., no other datasets have been used). Furthermore, we experimented with architectures ranging from Naïve Bayes (multinomial, complement and bernoulli variants), Support Vector Machines (SVMs with linear, polynomial and RBF kernels) to Logistic Regression which we tried to train using only the dataset provided by the SemEval-2022 Task 6 competition (Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy, 2022). The rest of the paper is organized as follows: section 2 briefly presents studies related to sarcasm

Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022), pages 970 - 977

detection, section 3 presents the dataset, the required pre-processing and plausible methods for it, section 4 resumes the results of the conducted experiments, with their interpretations, followed by section 5 with the conclusions.

2 Related work

This topic is a widely researched subject in recent years, evidenced in this competition at several workshops (e.g., SemEval-2017 Task 4: Using Sarcasm Detection for Enhancing Sentiment Classification and Quantification or SemEval-2018 Task 3: Irony Detection in English Tweets). Such a competition is challenging, especially since the problem of labeled data is time consuming and not cheap. Moreover, the automatic sarcasm detection depends on the annotation process, which always introduce some biases to the data. For the binary task, as in this case, there are many computational models to solve it or to capture the (actual) sarcasm or subcategories of it (John S. Leggitt and Raymond W. Gibbs Jr. , 2000) (Silviu Oprea and Walid Magdy, 2020). Thus, work on this topic was never followed by high results, as this problem is still debatable and text classification even for humans is very controversial and biased. It is a task that can be considered as sentiment analysis for which most of the authors used LSTM (Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, 2014), (Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy, 2016) or CNN (Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su, 2018) and (Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang, 2018). New approaches concentrate on using attention-based methods (Joshi et al., 2017), in particular transformer architectures such as BERT (Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina, 2018), RoBERTa, spanBERT (Amardeep Kumar, Vivek Anand, 2020) etc. These transformers are pre-trained on unlabeled data to be later fine-tuned for a variety of tasks like single sentence classification, sentence pair classification, etc. to understand role of context for sarcasm detection. Here, we used five machine learning approaches by trying to train the models based on this dataset solely (i.e., no other datasets have been used).

3 Dataset and Methods

This section is focused on two issues: Twitter dataset in English and two types of methods for classification task, both binary and multi-class. For the first type, we mostly tried classical machine learning approaches that resulted in satisfactory outcomes when evaluated on the training set. On the second one, a RNN with a few GRU layers was the architectural choice. At this moment, the results were modest, probably due to the lack of to much data (it is empirically known that neural networks require a high volume of data to work properly)

3.1 Dataset

The dataset that this competition provided consists of 3468 samples. Each sample is made of a short tweet and 8 binary columns which specify whether or not the text belongs to a certain class. The target column in the binary classification case is ”sarcastic” while the other 7 (sarcasm, irony, satire, understatement, overstatement, rhetorical_question, ambiguity) compose the outputs for the multi-class scenario. At a first glance, we can clearly see that the data is heavily unbalanced. However, this is pretty standard and conforms to the reality where, for example, only a few people are sarcastic, ironic or satirical and so on. Sure, data imputation methods could have been useful (for example, replacing the words of each text with the corresponding synonyms and adding the new texts to the dataset or doing some sort of oversampling on the positive label texts that are heavily lacking in examples) to make the set a little bit bigger but we chose to just stick to what was given! In the following picture (Figure 1), you can see a bar-chart of how the output column labels are distributed across the dataset.

One big inconvenience that we observed is the presence of many 0s in the target columns. This is difficult to solve since replacing nans with certain labels is not tractable. In the end, we went forward with the missing values. Before getting into the proper implementation details, we want to present an image of the overall architecture (Figure 2) so that every step is clear right from the start. The right arrows in the image define transitions from one step to another. A step title is written inside a cloud and the boxes under it are nothing but its definitory operations.

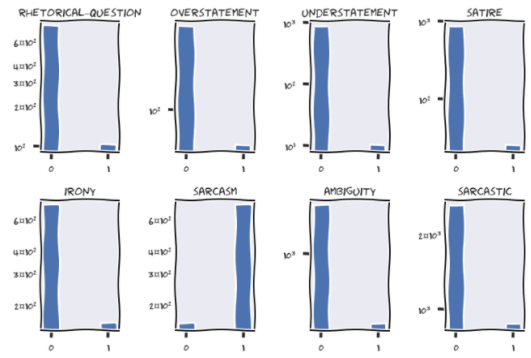

Figure 1: Bar charts of the labels distribution. The Y axis represents the counts for each value that the labels take.

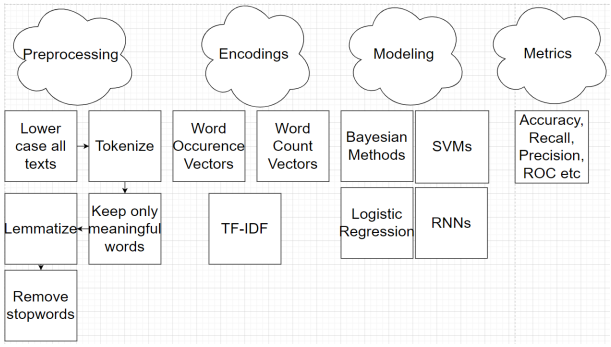

Figure 2: The architecture is made out of 4 parts. The separation between them is made through arrows.

3.2 Methods

3.2.1 Pre-processing

Before talking about the machine learning models, we will explain the pre-processing step that had to be done in order to ”clear” the data. First, we lowered all the training texts and tokenized them into words. Since many compound words could exist (e.g. ”state-of-the-art” or ”equality”; note that these are not actual words from the set) the natural thing to do is to split them by space. After that, for each sample, we are left with lists of words only, hence getting rid of the non alphanumeric words is advisable (the punctuation marks do not add any real value to our task). Another requirement is the lemmatization of the words. We want the root forms of our words in order not to consider multiple derivations of the same word as different (e.g. ”goes” should be the same with ”go”, ”going” or ”gone” and so on). Now, as a final touch, we eliminate the stopwords (the stopwords list is taken from the corpus class of the nltk library) since they don’t add any value to our models. We also built a vocabulary which contains all the unique words of the corpus and this will help us when constructing the numerical representations. Given the processed texts, we can start the well known one hot, doc-term and tf-idf encodings. For each one of them, we define a matrix where the rows represent the indexes of the texts and the columns are nothing but the words from our vocabulary. In the case of one-hot representation, one cell is either 1 or 0, noting the apppearance of a word (column) in the text (row). Alternatively, doc-term counts the number of appearances of a word in a text while tf-idf is built on top of doc-term and gives some scores according to the frequencies of the terms. All of these encodings were used with different machine learning methods.

3.2.2 Classical Machine Learning Approaches

As a first approach to classification we tried bayesian methods. We chose them because they are fast to build and have low variance. As another plus, the performance is very good with both small training sets, as well as big ones. Having a reputation as a bad estimator, but a decent classifier, we decided that it’s a good starting point (please note that the previous statements can be deducted from the books (Tom M. Mitchell, 1997), the Bayesian Learning Chapter and (Christopher Bishop, 2006), the Graphical Models chapter). Throughout time, NB has been the most used one when it comes to NLP tasks and we thought that our numerical representations could work very well on it. Bernoulli Naïve Bayes (BNB), Multinomial Naïve Bayes (MNB) and Complement Naïve Bayes (CNB) are the options we have through the scikit-learn library. BNB deals with binary features, hence one-hot encoding is the only choice here. Multinomial Naïve Bayes works with word counts or word scores, therefore doc-term and tf-idf representations are the way to go in this direction. As a last resort, we also tried a CNB because, according to the scikit documentation (Complement Naive Bayes documentation of scikit), CNB is an adaptation of the standard MNB algorithm that is particularly suited for imbalanced data sets, just like ours. The inventors of CNB show empirically that the parameter estimates for CNB are more stable than those for MNB. Further, CNB regularly outperforms MNB (often by a considerable margin) on text classification tasks (Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger). It is well known that logistic regression is the discriminative corre-

spondent of Naïve Bayes. Moreover, if the conditional independence supposition of NB is true and the amount of training data tends to infinite, the 2 models should give similar results. Therefore, we tried to fit a logistic regression on our data to check the previous statement (these statements are based on the Generative and discriminative classifiers (Tom M. Mitchell) book chapter). Moving onto the next level, support vector machines (SVMs) come into play. Considering that each numerical representation has lots of features (words) and only a few lines (samples), the dual form is preferred, since the complexity is O(rows) (rows are the number of samples). Besides this, we work with the soft-margin SVM because it allows us to avoid overfitting through the slack variable. When it comes to kernels, we had 3 options. The first one we chose is the linear kernel (a.k.a the "no kernel") because in a very high dimensional space as ours, it's plausible to find a good separating hyperplane. After that, we decided to try a polynomial kernel as well. This one maps our data into a higher dimension, hoping to find a better suited hyperplane. The degrees used were 2 and 3 and we compared the results with the previous model. In the end, the RBF kernel seemed to be the natural step forward because it's the only one that is guaranteed to find a separating hyperplane by mapping the features into an infinite dimensional space. The hyper-parameters gamma and C (the weight of the slack variables) were chosen with a randomized grid search (no specific random seed, we just used the scikit default option) with cross validation. In other words, different values from 2 exponential distributions have been generated to each hyper-parameter and then, a SVM was applied. The one that gave the best results at cross validation was kept.

3.2.3 Deep Learning Approaches

Even though it was not the main target of the project, we chose to take a closer look at the other classes of the dataset and fit a neural network that could be the starting point of some future work. On the multi-class classification, as we've said, we fitted a RNN. However, this was not a classic RNN, since those lose to much early information. To mediate this problem, we thought that GRU hidden layers (HL) are a good option. Moreover, due to the lack of data, we had to stick to a number of 3 hidden layers only. Each one of them has 10 neurons. The output layer is, of course, a 7 neurons dense layer with softmax activation. To avoid overfitting, we also added a dropout of 10% on each HL. When it comes to the input layer, the situation is a bit different. Given that each text sample has a different number of words, some padding had to be done. Therefore, we calculated the length of the longest words sequence sample and padded the rest of the samples with 0 until we would obtain that maximum length for each training observation. The last requirement was a word2vec matrix where the indexes represent a word and the columns are the dimensions of our numerical mappings. This number of dimensions was chosen by us to be 5 because we don't have that many words. To speed up the training process, a batch normalization layer was used immediately after the input layer. The results were not the best, as we will see in the next section. We definitely needed more training data.

4 Results and Interpretations

We would like to make it clear that the results have been obtained on the training set here, since we didn't have a test set available at the time we were working on the project. Moreover, the results should be viewed with caution since the models have been trained on a small sized data set that has not to many samples, hence not to many words. A big problem arises: what do we do when the future testing samples don't contain words we've seen before? To answer this, one could simply drop the unseen words and focus on those seen only. Moving on, the main metrics we have used throughout the project are: recall, precision, F-score, balanced accuracy, accuracy and the ROC graph. It's worth noticing that the metrics have been evaluated on all the numerical representations we have mentioned before. We will start with the Naïve Bayes algorithms since these were the ones that surprisingly produced the best results (Table 1).

Bear in mind that the results have obtained on a multinomial variant with doc-term numerical representation. We can clearly see that there is a difference between the accuracy and the balanced accuracy of 8%. This is quite a big percentage, but not as big as other models resulted. Of course, the standard accuracy is not very relevant since the data is imbalanced, hence one should pay attention to the balanced accuracy more. When it comes to precision and recall on the negative labels (i.e., the not sarcastic texts), the system is very precised. Both a high sensitivity and high precision are what

you would normally want from a system. In other words, the non-sarcastic texts are very well classified. On the positive labels, although not as big, the rate on recall and precision maintains at a strong above 80% percentage. Judging by the F-scores, we are pleased with the result, however, as we've said in the previous chapter, Naïve Bayes is a good classifier, but a bad estimator, so even though the results are great, the independence supposition in this case is very general and wide and tricks the statistical principles. The results for the rest of the other NB variants are given in Table 2 and Table 3.

Moving on to logistic regression, we can see that the results are really far away from NB. Therefore, we can definitely say that the Naïve Bayes independence supposition here is strongly untrue. We can see that the recall on the positive labels is very small, but the precision is almost close to 100%. In other words, among all the texts that are positive in reality, only a small percentage of them are classified as positive, but those that are predicted positive, are done so with very high certainty. So, we could say that someone who is interested in getting a very precised sarcastic prediction should choose this model. Also, an interesting observation that we can draw from all the analysed algorithm results is that a high recall on the positive class leads to a high precision on the negative class and alternatively a high precision on the positive class leads to a high recall on the negative class (Table 4).

The support vector machines are the ones that behaved the most poorly among the tried models. This is somehow weird since most of the articles out there indicate SVMs as the best option for binary classification in NLP when having little data. As mentioned before, we tried different kernel variants, but nothing good came up, unfortunately.

The polynomial kernel seems to have given the best results, but these are weak. Table 5 and Table 6 are associated to the linear and polynomial variants. Judging by the F-scores and balanced accuracy, we can easily conclude that the bayesian methods are behaving way better. What we found to be very curious is the fact that the randomized grid search didn't give any good SVM, even after 200 iterations and generous exponential distributions for "C" and "gamma" hyper-parameters. For this reason, we chose to skip showing the results for the RBF kernel. In the end, we will look at the ROC curve (Figure 9) that sums up all the models

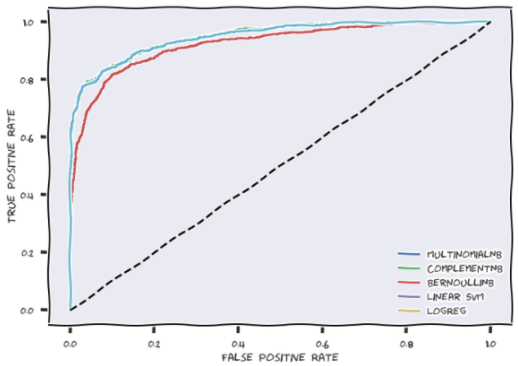

Figure 3: ROC curve

that we fitted in our project. Again, the curve with the highest area under it is the Multinomial Naïve Bayes one. The other curves just seem to overlap, firmly bellow the best one!

5 Conclusion

For sarcasm classification task (binary or multi-class), a small dataset is a really challenging. The classical machine learning approaches we have presented so far cannot answer convenient to it. This lack of data is a big downside because it means that not many words are numerically encoded, hence having the same vocabulary on a testing set is absolutely mandatory. In practice, it is very rarely. We may conclude that the best model is the Multinomial Naïve Bayes one, but again, it is not guaranteed that it has not overfitted the training data. In the future, some more attention could be payed to the neural network if data imputation is done and the dataset reaches a reasonable number of samples.

References

Amardeep Kumar, Vivek Anand. 2020. Transformers on Sarcasm Detection with Context. *Proceedings of the Second Workshop on Figurative Language Processing*, pages 88–92.

C. Van Hee, E. Lefever, and V. Hoste. 2018. Exploring the fine-grained analysis and automatic detection of irony on Twitter. *Lang Resources Evaluation Vol. 52, pages 707–731.*

Christopher Bishop. 2006. Pattern Recognition and Machine learning. *Springer, New York.*

Complement Naive Bayes documentation of scikit. https://scikit-learn.org/stable/modules/naive_bayes.html#complement-naive-bayes.

Dan Alexandru and Daniela Gîfu. 2020. Tracing humor in Edited News Headlines. *Ludic, Co-design and Tools Supporting Smart Learning Ecosystems and Smart Education, pages 187–196. Springer Singapore.*

Devlin, Jacob and Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *ICLR*.

Gîfu, Daniela. 2010. The Discourse of the Written Press and the Violence of Symbols. *PhD thesis, Faculty of Philosophy and Political Studies,"Alexandru Ioan Cuza" University of Iaşi*.

Gifu Daniela, Samson Mihai. 2021. FII FUNNY at SemEval-2021 Task 7: HaHackathon: Detecting and rating Humor and Offense. *Proceedings of the 15th International Workshop on Semantic Evaluation, (SemEval-2021), ACL, Bangkok, Thailand, pages 1226–1231, DOI: 10.18653/v1/2021.semeval-1.174*.

Ibrahim Abu Farha, Silviu Oprea, Steven Wilson, and Walid Magdy. 2022. 2022. SemEval-2022 Task 6: iSarcasmEval, Intended Sarcasm Detection in English and Arabic. *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022). Association for Computational Linguistics*.

Jason D. M. Rennie, Lawrence Shih, Jaime Teevan, David R. Karger. Tackling the Poor Assumptions of Naive Bayes Text Classifiers. Massachusetts Institute of Technology.

John S. Leggitt and Raymond W. Gibbs Jr. . 2000. Emotional reactions to verbal irony.

Reyes, Antonio and Rosso, Paolo and Buscaldi, Davide. 2012. From humor recognition to irony detection: The figurative language of social media. *Data & Knowledge Engineering*, 74:1–12.

Silviu Oprea and Walid Magdy. 2020. The effect of sociocultural variables on sarcasm communication online. *Proceedings of the 23rd ACM Conference on Computer-Supported Cooperative Work and Social Computing (CSCW).*

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018. Disan: Directional self-attention network for rnn/cnn-free language understanding. *AAAI*.

Tom M. Mitchell. http://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf.

Tom M. Mitchell. 1997. Machine Learning,. *McGraw Hill Education, New York*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in between. *ACL*, pages 1010–1020.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1480–1489.

	Precision	Recall	F1-Score	Support	Overall
0	0.90	0.99	0.94	2601	-
1	0.97	0.66	0.78	867	-
balanced accuracy	-	-	-	3468	0.8254
accuracy	-	-	-	3468	0.9094
macro avg	0.93	0.83	0.86	3468	-
weighted avg	0.92	0.91	0.90	3468	-

Table 1: Multinomial NB with doc-term encoding

	Precision	Recall	F1-Score	Support	Overall
0	0.89	0.99	0.94	2601	-
1	0.98	0.65	0.78	867	-
balanced accuracy	-	-	-	3468	0.8208
accuracy	-	-	-	3468	0.9077
macro avg	0.93	0.82	0.86	3468	-
weighted avg	0.91	0.91	0.90	3468	-

Table 2: Multinomial NB with one hot encoding

	Precision	Recall	F1-Score	Support	Overall
0	0.76	1.00	0.87	2601	-
1	1.00	0.07	0.13	867	-
balanced accuracy	-	-	-	3468	0.5340
accuracy	-	-	-	3468	0.7670
macro avg	0.88	0.53	0.50	3468	-
weighted avg	0.82	0.77	0.68	3468	-

Table 3: Multinomial NB with tf-idf encoding

	Precision	Recall	F1-Score	Support	Overall
0	0.96	1.00	0.98	2601	-
1	0.99	0.87	0.93	867	-
balanced accuracy	-	-	-	3468	0.9323
accuracy	-	-	-	3468	0.9653
macro avg	0.98	0.93	0.95	3468	-
weighted avg	0.97	0.97	0.96	3468	-

Table 4: Logistic Regression Results

	Precision	Recall	F1-Score	Support	Overall
0	0.85	1.00	0.92	2601	-
1	1.00	0.49	0.66	867	-
balanced accuracy	-	-	-	3468	0.7452
accuracy	-	-	-	3468	0.8722
macro avg	0.93	0.75	0.79	3468	-
weighted avg	0.89	0.87	0.86	3468	-

Table 5: SVM with linear kernel results

	Precision	Recall	F1-Score	Support	Overall
0	0.87	1.00	0.93	2601	-
1	1.00	0.54	0.70	867	-
balanced accuracy	-	-	-	3468	0.7710
accuracy	-	-	-	3468	0.8855
macro avg	0.93	0.77	0.82	3468	-
weighted avg	0.90	0.89	0.87	3468	-

Table 6: SVM with polynomial kernel results