CPH: Sentiment analysis of Figurative Language on Twitter #easypeasy #not

Sarah McGillion Héctor Martínez Alonso Barbara Plank
University of Copenhagen, Njalsgade 140, 2300 Copenhagen S, Denmark
zhg159@alumni.ku.dk,alonso@hum.ku.dk,bplank@cst.dk

Abstract

This paper describes the details of our system submitted to the SemEval 2015 shared task on sentiment analysis of figurative language on Twitter. We tackle the problem as regression task and combined several base systems using stacked generalization (**?**). An initial analysis revealed that the data is heavily biased and a general sentiment analysis system (GSA) performs poorly on it. However, GSA proved helpful on the test data, which contains an estimated 25% non-figurative tweets. Our best system, a stacking system with back off to GSA, ranked 4th on the final test data (Cosine 0.661, MSE 3.404).[1]

1 Introduction

Sentiment analysis (SA) is the task of determining the sentiment of a given piece of text. The amplitude of user-generated content produced every day raises the importance of accurate automatic sentiment analysis, for applications ranging from, e.g., reputation analysis (**?**) to election results prediction (**?**). However, figurative language is pervasive in user-generated content, and figures of speech like irony, sarcasm and metaphors impose relevant challenges for a sentiment analysis system usually trained on literal meanings. For instance, consider the following example:[2] *@CIA We hear you're looking for sentiment analysis to detect sarcasm in Tweets. That'll be easy! #SLA2014 #irony*. Irony or sarcasm does not result always in the exact opposite sentiment and therefore it is not as simple as just inverting the scores from a general SA system. Only few studies have attempted SA on figurative language so far (**?**; **?**).

The prediction of a fine-grained sentiment score (between -5 and 5) for a tweet poses a series of challenges. First of all, accurate language technology on tweets is hard due to *sample bias*, i.e., collections of tweets are inherently biased towards the particular time (or way, cf. §**??**) they were collected (**?**; **?**). Secondly, the notion of figurativeness (or its complementary notion of literality) does not have a strong definition, let alone do irony, sarcasm, or satire. As pointed out by **?**), "there is not a clear distinction about the boundaries among these terms". Yet alone attaching a fine-grained score is far from straightforward. In fact, the gold standard consists of the average score assigned by humans through crowdsourcing reflecting an uncertainty in ground truth.

2 Data analysis

The goal of the initial data exploration was to investigate the amount of non-figurativeness in the train and dev data. Our analysis revealed that 99% of the training data could be extracted using a simple heuristic: a regular expression decision list, hereafter called Tweet Label System (TLS), to split the training data into different key-phrase subgroups. The system searches for the expression in a tweet and then assigns a label in a cascade fashion following the order in Table **??**. Table **??** lists the 14 possible label types (plus NONE), their associated expressions along with the support for each category in the training data. Table **??** shows that only a small

[1] After submission time we discovered a bug in ST2,which means that the results on the official website are of the GSA and not of the stacking system with back off.

[2] From the train data, label: -1.24; GSA prediction: +5

fraction of the train and dev data could not be associated to a subgroup and it can be seen that the final test data was estimated to have a very different distribution with 25% of tweets presumably containing literal language use.

Dataset	Train	Trial	Test
Instances	7988	920	4000
% Non-figurative	1%	7%	25%

Table 1: Retrieved instances in each data set and estimated amount of non-figurativeness.

Since there are obvious subgroups in the data, our hypothesis is that this fact can be used to construct a more informed baseline. In fact (§ **??**), simply predicting the mean per subgroup pushed the constant mean baseline performance considerably (from 0.73 to 0.81 Cosine, compared to random 0.59).

Figure **??** plots predicted scores (ridge model, §**??**) of three subgroups against the gold scores on the trial data. It can be seen that certain subgroups have similar behaviour, 'sarcasm' has a generally negative cloud and the model performs well in predicting these values, while other groups such as 'SoToSpeak' have more intra-group variance.

Label	Expression	Support	Label	Expression	Support
Sarcasm	#sarcas	2139	SoToSpeak	so to speak	135
Irony	#iron(y ic)	1444	Proverbial	proverbial	22
Not	#not	3601	JustKidding	#justkidding	-
Literally	literally	344	Not2	not	29
Virtually	virtually	8	about	about	8
YeahRight	#yeahright	47	Oh	oh	3
OhYouMust	Oh.*you	2	NONE	-	92
asXas	as .* as	83			

Table 2: Tweet Label Type and Expression.

The effect of a general sentiment system

The data for this task is very different from data that most lexicon-based or general sentiment-analysis models fare best on. In fact, running a general sentiment classifier (GSA) described in **?**) on the trial data showed that its predictions are actually anti-correlated with the gold standard scores for the Tweets in this task (cosine similarity score of -0.08 and MSE of 18.62). We exploited these anti-correlated results as features for our stacking systems (cf. § **??**). Figure **??** shows the distributions of the gold scores and GSA predictions for the trial

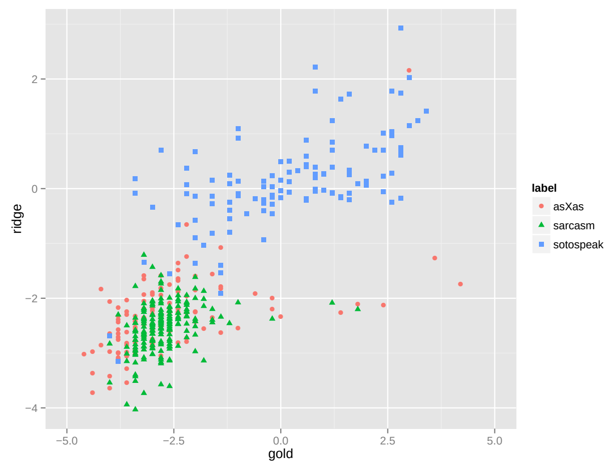

Figure 1: Label Plots for RR predictions.

data. It shows that the gold distribution is skewed with regards to the number of negative instances to positives, while the GSA predicts more positive sentiment.

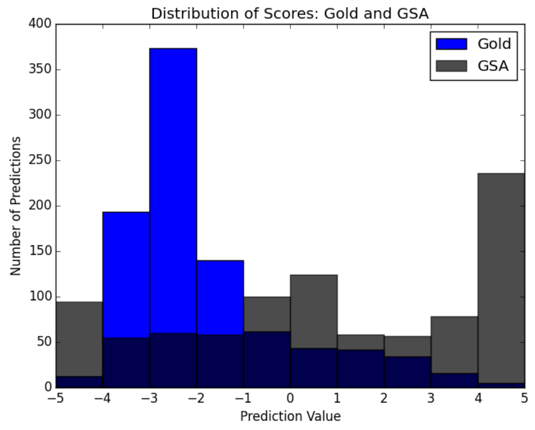

Figure 2: Distribution of Gold Scores and GSA Predictions for Trial Data.

3 System Description

We approached the task (**?**) as a regression task (cf. §**??**), combined several systems using stacking (§ **??**), and rely on features without POS, lemma or explicit use of lexicons, cf. § **??**.

3.1 Single Systems

Ridge Regression (RR) A standard supervised ridge regression model with default parameters.[3]

PCA_GMM Ridge Regression (GMM) A ridge regression model trained on the output of unsupervised induced features, i.e., a Gaussian Mixture Models (GMM) trained on PCA of word n-grams. PCA was used to reduce the dimensionality to 100, and GMM under the assumption that the data was sampled from different distributions of figurative language, k Gaussians were assumed (here $k = 12$).

Embeddings with Bayesian Ridge (EMBD) A Bayesian Ridge Regressor learner with default parameters trained on only word embeddings. A corpus was build from the training data and a Tweet collection sampled with the labels from the TLS and some key words identified from the data e.g. 'speak'. This resulted in a total of 3.7 million tweets and 67 million tokens. For details on how the word embeddings were built see §**??**.

3.2 Ensembles

We developed two stacking systems (**?**), *Stacking System 1* (ST1) and *Stacking System 2: Stacking with Back Off* (ST2). The systems used for these are shown in Table **??** and the Meta Learner used for both stacking systems is Linear Regression.

The systems used in ST1 and ST2 are not the only differences between the two. ST2 uses the TLS to identify the subgroup that each tweet belongs to. For any Tweet with the NONE subgrouping, the system would back off to the predictions from the GSA. We built ST2 as a system that is not limited to sentiment analysis for a small subsection of language, the phenomenon of figurative language, but is applicable in situations covering many types of Tweets including those in which literal language is used.

Single System / Stacking System	ST1	ST2
RR	X	X
GMM	X	
EMBD		X
GSA	X	X

Table 3: Systems in Ensemble Setups.

[3] `http://scikit-learn.org/`

3.3 Features

This section describe the features we used for the models in §**??**. Table **??** indicates the type of features used for the single models. Punctuation was kept as its own lexical item and we found removing stopwords and normalizing usernames to '@USER' increased performance and as such the preprocessing methods are the same across the models. Features were set on the trial data.

Word N-Grams Systems use different n-grams as features. In RR counts of 1 and 5 word grams, in GMM binary presence of 1,2, and 3 word grams.

Uppercase Words Counts of the numbers of word in a Tweet with all uppercase letters.

Punctuation Contiguous sequences of question, exclamation, and question and exclamation marks.

TLS Label The subgrouping label from the TLS.

Word Embeddings Parameters for word embeddings:[4] 100 dimensions, 5 minimum occurrences for a type to be included in the model, 5 word context window and 10-example negative sampling. Each tweet was represented by 100 features that represented the average of all the embeddings of the content words in the tweet.

Features/Systems	RR	GMM	EMBD
Word N-grams	X	X	
Uppercase	X		
Punctuations	X		
TLS Label	X		
Word Embeddings			X

Table 4: Features used in Single Models.

4 Results

4.1 Constant baselines & Single Systems

We implemented the Mean, Mode, Median, Random and TSL (§**??**) baseline systems. TSL is the hardest baseline, and RR is the only system that beats it.

[4] `https://code.google.com/p/word2vec/`

System	Cosine	MSE
TLS	**0.81**	**2.34**
Mean	0.73	3.13
Mode	0.73	3.13
Median	0.73	3.31
Random	0.59	5.17
RR	**0.88**	**1.60**
GMM	0.79	2.55
EMB	0.78	2.64

Table 5: Baseline and Single Systems On Trial Data.

4.2 Results Stacking Systems

The performance of the stacking systems on the trial data can be seen below in Table **??**. ST2 did not perform well on the trial data although a reason for this is that only 7% of the trial data was found as not belonging to a known figurative type of tweet.

System	Cosine	MSE
ST1	0.86	1.88
ST2	0.79	2.57

Table 6: Stacking Model Results on Trial Data.

4.3 Final Results

Three models were submitted for final evaluation on the test data. The three models were RR, ST1, and ST2. For the final results we scaled back values outside the range [-5,5] to the nearest whole number in range. Tables **??** and **??** show the results for our systems on the final dataset and the performance of the overall winning system for the task (CLAC) . Table **??** shows the overall cosine similarity and MSE for the systems on the test data and Table **??** shows the breakdown of the cosine similarity for the systems on the different parts of language. It is interesting to note that the performance of ST2 on the 'Other' type of language is identical as the performance for CLAC, this is also the best cosine similarity score 'Other' out of all submissions.

[5]The numbers in bold indicate the best performance among our systems, underlined indicates the best performance between any of our systems and the winning system.

System	Test Cosine	Test MSE
RR	0.625	3.079
ST1	0.623	**3.078**
ST2	**0.661**	3.404
CLAC	0.758	2.117

Table 7: Submission System Test Results.[5]

System	Overall	Sarcasm	Irony	Metaphor	Other
RR	0.625	0.897	0.886	0.325	0.218
ST1	0.623	**0.900**	**0.903**	0.308	0.226
ST2	**0.661**	0.875	0.872	**0.453**	**0.584**
CLAC	0.758	0.892	0.904	0.655	0.584

Table 8: Cosine Test Results Breakdown.

4.4 The case for regression

Regression is less usual in NLP than classification. In this section we show that for this data, it is desirable to use regression. It incorporates the relation between the labels instead of treating them as orthogonal—in addition to the advantage of keeping the decimal precision in the target variable when training. We ran classification experiments for this task but found that the best classification system's[6] performance (Cosine 0.82, MSE 2.51) is still far from the RR model (0.88,1.60).

5 Conclusions

We tested three systems for their abilities to analyse sentiment on figurative language from Twitter. Our experiments showed that a general SA system trained on literal Twitter language was highly anti-correlated with gold scores for figurative tweets. We found that for certain figurative types, sarcasm and irony, our system's predictions for these phenomena faired well. Our system did not explicitly use a lexicon to define the sentiment of a tweet, but instead used machine learning and strictly corpus-based features (no POS or lemma) to place us 4th in the task. More effort may be needed to discriminate metaphorical from literal tweets to build a more robust system, although, even for humans the sentiment of tweets is hard to judge.

[6]Decision Tree with 7 classes and using the minimum score for instances in the classes in the training data to convert for class labels to scores.