WASSA-2017 Shared Task on Emotion Intensity

December 19, 2017

Task: Given a tweet and an emotion X, determine the intensity or degree of emotion X felt by the speaker – a real-valued score between 0 and 1.

—

Evaluation: For each emotion, systems are evaluated by calculating the Pearson Correlation Coefficient with Gold ratings. The correlation scores across all four emotions will be averaged to determine the bottom-line competition metric by which the submissions will be ranked.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Данные

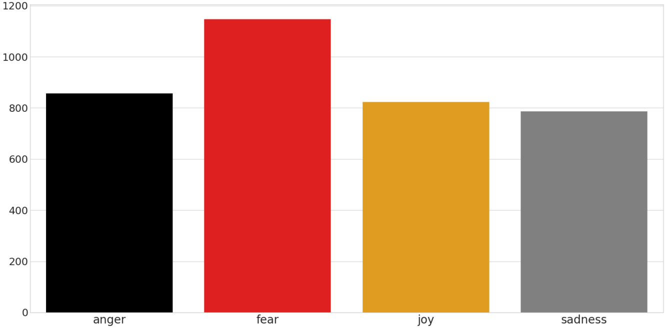
Четыре отдельных датасета для каждой эмоции:

- anger (злость)
- fear (страх)
- sadness (грусть)
- joy (радость)

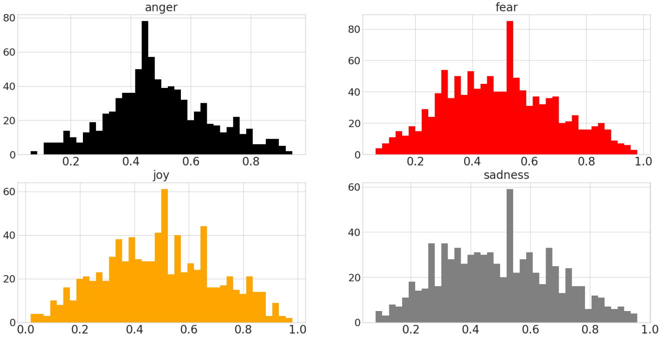
В тестовых данных известна эмоция, нужно предсказать интенсивность.

	id	text	emotion	intensity
0	10000	How the fu*k! Who the heck! moved my fridge!... should I knock the landlord door. #angry #mad ##	anger	0.938
1	10001	So my Indian Uber driver just called someone the N word. If I wasn't in a moving vehicle I'd have jumped out #disgusted	anger	0.896
2	10002	@DPD_UK I asked for my parcel to be delivered to a pick up store not my address #fuming #poorcustomerservice	anger	0.896

Размерность датасетов



Гистограмма интенсивности для каждой эмоции



Самые популярные слова в датасетах

Дубликаты

```
duplicate_example = pd.merge(train_anger[~train_anger['duplicate'].isnull()],
        train_anger[['id','text','intensity']],
        how='left',
        left_on='duplicate', right_on='id',
        suffixes=('_left', '_right'))[['text_left','intensity_left','text_right','intensity_right']]
```

```
duplicate_example.head(6)
```

	text_left	intensity_left	text_right	intensity_right
0	So my Indian Uber driver just called someone the N word. If I wasn't in a moving vehicle I'd have jumped out #disgusted	0.896	So my Indian Uber driver just called someone the N word. If I wasn't in a moving vehicle I'd have jumped out #disgusted #offended	0.729
1	@DPD_UK I asked for my parcel to be delivered to a pick up store not my address #fuming #poorcustomerservice	0.896	@DPD_UK I asked for my parcel to be delivered to a pick up store not my address #poorcustomerservice	0.625
2	so ef whichever butt wipe pulled the fire alarm in davis bc I was sound asleep #pissed #angry #upset #tired #sad #tired #hangry ######	0.896	so ef whichever butt wipe pulled the fire alarm in davis bc I was sound asleep #pissed #upset #tired #sad #tired #hangry ######	0.771
3	Don't join @BTCare they put the phone down on you, talk over you and are rude. Taking money out of my acc willynilly! #fuming	0.896	Don't join @BTCare they put the phone down on you, talk over you and are rude. Taking money out of my acc willynilly!	0.604
4	My blood is boiling	0.875	@ArizonaCoyotes not to mention the GRA guy stops me but let's the 2 ppl in front of me go. WTF. My blood is boiling.	0.854
5	When you've still got a whole season of Wentworth to watch and a stupid cunt in work ruins it for us 😭😭 @__KirstyGA #raging #oldcunt	0.875	When you've still got a whole season of Wentworth to watch and a stupid cunt in work ruins it for us 😭😭 @__KirstyGA #oldcunt	0.625

Сарказм

	id	text	emotion	intensity
50	10050	Im so angry 😂🙃	anger	0.75

TF-IDF

Применим к тексту TF-IDF и на полученных признаках обучим градиентный бустинг.

	params	mean_test_score
0	{'max_depth': 6, 'n_estimators': 100}	0.552942
1	{'max_depth': 6, 'n_estimators': 150}	0.562530
2	{'max_depth': 6, 'n_estimators': 200}	0.566330
3	{'max_depth': 6, 'n_estimators': 250}	0.566890
4	{'max_depth': 6, 'n_estimators': 300}	0.568344
5	{'max_depth': 6, 'n_estimators': 350}	0.567639

NRC Hashtag Emotion Association Lexicon (HE)

Для каждого слова получим вектор эмоций размерности 8.
Предложение представим как усреднение по всем словам.

	emotion	term	score
0	anticipation	crae	2.237478
1	anticipation	#mycolour	2.237478
2	anticipation	#vigilance	2.237478

```
pd.DataFrame(gs.cv_results_)[['params','mean_test_score']]
```

	params	mean_test_score
0	{'max_depth': 2, 'n_estimators': 50}	0.552387
1	{'max_depth': 2, 'n_estimators': 100}	0.550712
2	{'max_depth': 2, 'n_estimators': 200}	0.540669
3	{'max_depth': 2, 'n_estimators': 300}	0.534076
4	{'max_depth': 3, 'n_estimators': 50}	0.566023
5	{'max_depth': 3, 'n_estimators': 100}	0.556676
6	{'max_depth': 3, 'n_estimators': 200}	0.548189
7	{'max_depth': 3, 'n_estimators': 300}	0.542666

NRC Affect Intensity Lexicon (AI)

Для каждого слова получим вектор эмоций размерности 4.
Предложение представим как усреднение по всем словам.

	term	score	AffectDimension
0	outraged	0.964	anger
1	brutality	0.959	anger
2	hatred	0.953	anger

```
pd.DataFrame(gs.cv_results_)[['params','mean_test_score']]
```

	params	mean_test_score
0	{'max_depth': 2, 'n_estimators': 50}	0.266638
1	{'max_depth': 2, 'n_estimators': 100}	0.289034
2	{'max_depth': 2, 'n_estimators': 200}	0.308386
3	{'max_depth': 2, 'n_estimators': 300}	0.329125
4	{'max_depth': 3, 'n_estimators': 50}	0.295239
5	{'max_depth': 3, 'n_estimators': 100}	0.325313
6	{'max_depth': 3, 'n_estimators': 200}	0.339517
7	{'max_depth': 3, 'n_estimators': 300}	0.346879

AFINN

Для каждого слова получим сентимент. Предложение представим как сумму сентиментов.

	term	score
0	abandon	-2
1	abandoned	-2
2	abandons	-2

	params	mean_test_score
0	{'max_depth': 3, 'n_estimators': 50}	0.380138
1	{'max_depth': 3, 'n_estimators': 100}	0.377060
2	{'max_depth': 3, 'n_estimators': 200}	0.376253
3	{'max_depth': 3, 'n_estimators': 300}	0.376202

GLOVE

	params	mean_test_score
0	{'max_depth': 3, 'n_estimators': 100}	0.535247
1	{'max_depth': 3, 'n_estimators': 150}	0.531898
2	{'max_depth': 3, 'n_estimators': 200}	0.527017

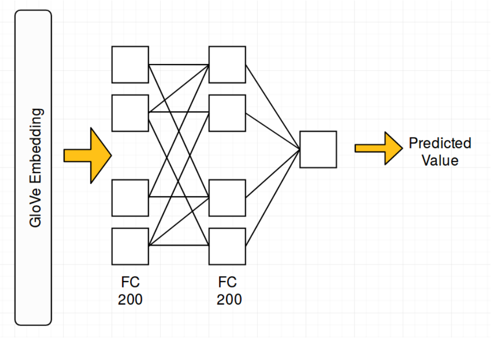
HE-AI-AFINN-GLOVE

Используем все три лексикона вместе с GloVe embedding

```
for emotion in main_emotions:
    get_score(HE_AI_AFINN_GLOVE_TFIDF_features, emotion, 250)
```

```
emotion:  anger
dev:  (0.63486529452719065, 0.58674616659130574, 0.30587742199660189, 0.11427440048267601)
test:  (0.65697183791102332, 0.63746346917179952, 0.48133252757713418, 0.46621846793702326)
emotion:  fear
dev:  (0.62462101442565898, 0.5989504122076863, 0.59539017174914566, 0.5832430693774453)
test:  (0.68557640655072993, 0.6590311230539514, 0.55178771540929517, 0.50121771642496027)
emotion:  joy
dev:  (0.71318021409181021, 0.70434058893528673, 0.58250645895220243, 0.55615817481744922)
test:  (0.67334035094369304, 0.67003299299756947, 0.45214426102652522, 0.44990347206878278)
emotion:  sadness
dev:  (0.48623640152260184, 0.47453660577008538, 0.25726979752336437, 0.24226944359024849)
test:  (0.65457801161145202, 0.65182800229517746, 0.45491368120652187, 0.43724048220430028)
```

Лексиконы покрывают не все слова. При этом слова близкие по эмоциональному оттенку, скорее всего будут лежать в пространстве эмбедингов где-то рядом.
Используем нейронную сеть для расширения лексикона.



Расширение лексикона

```
afinn_word2index = dict(zip(afinn_lexicon['term'],list(afinn_lexicon.index)))
```

```
X = np.zeros((len(afinn_word2index), 400))
Y = afinn_lexicon['score'].values

for word, index in afinn_word2index.items():
    if word in word2vec:
        X[index,:]=word2vec[word]
```

```
def make_model(optimizer='adam', loss='mse', activation='relu', layer_size=200, use_third_layer=False):
    model = Sequential()
    model.add(Dense(layer_size, activation=activation, input_shape=(400,)))
    model.add(Dense(layer_size, activation =activation))
    if use_third_layer:
        model.add(Dense(25, activation=activation))
    model.add(Dense(1))
    model.compile(optimizer, loss)

    return model
```

HE AI AFINN GLOVE TFIDF EXTENDED

Собрав все вместе, получаем следующие результаты.

```
HE_AI_AFINN_GLOVE_TFIDF_EXTENDED_features = {}
for emotion in main_emotions:
    HE_AI_AFINN_GLOVE_TFIDF_EXTENDED_features[emotion] = [np.hstack([HE_features[emotion][i],
                                                          AI_features[emotion][i],
                                                          AFINN_features[emotion][i],
                                                          glove_features[emotion][i],
                                                          TF_IDF_features[emotion][i].toarray(),
                                                          extended_features[emotion][i]
                                                         ]) for i in range(3)]
```

```
for emotion in main_emotions:
    get_score(HE_AI_AFINN_GLOVE_TFIDF_EXTENDED_features, emotion, 250)
```

```
emotion:  anger
dev:  (0.6195997855766123З, 0.5597206266018026б, 0.3263307957997541б, 0.1669839948012535)
test:  (0.68003213530182127, 0.66517941554000848, 0.50631076212682113, 0.49703304138486004)
emotion:  fear
dev:  (0.62756769977987037, 0.58699967239441297, 0.59429330031598537, 0.57417078711372194)
test:  (0.69130447589702382, 0.66759021621573078, 0.55643618909279147, 0.50620116763701539)
emotion:  joy
dev:  (0.73110455983548051, 0.72605868027532428, 0.63584760790963379, 0.63296048790027437)
test:  (0.68308106431810189, 0.68241787389621622, 0.46408315832623098, 0.46421985555687106)
emotion:  sadness
dev:  (0.52336706756955909, 0.52002417758800401, 0.23091339814682707, 0.24573397087345944)
test:  (0.67093316071846076, 0.66779950784249564, 0.43142882129481169, 0.41678040153735219)
```

Данная моделька перебивает baseline от авторов задачи.

- имя пользователя в твите заменяем на username.
- хештеги, которых нет в word2vec, заменяем на #hashtag
- замена смайлика на слово описывающее смайлик не сработало
- восклицательные знаки сохраняем

```python
def text_to_wordlist(text, remove_stopwords=False, stem_words=False, w2v=None):
    text = re.sub(r"@\w{1,15}", "USERNAME", text)
    text = re.sub(r"[^A-Za-z0-9^,!.#\/'+-=]", " ", text)
    text = re.sub(r"what's", "what is ", text)
    text = re.sub(r"\'s", " ", text)
    text = re.sub(r"\'ve", " have ", text)
    text = re.sub(r"can't", "cannot ", text)
    text = re.sub(r"n't", " not ", text)
    text = re.sub(r"i'm", "i am ", text)
    text = re.sub(r"\'re", " are ", text)
    text = re.sub(r"\'d", " would ", text)
    text = re.sub(r"\'ll", " will ", text)
    text = re.sub(r",", " ", text)
    text = re.sub(r"\.", " ", text)
    text = re.sub(r"!", " !", text)
    text = re.sub(r"\/", " ", text)
    text = re.sub(r"\^", " ^ ", text)
    text = re.sub(r"\+", " + ", text)
    text = re.sub(r"\-", " - ", text)
    text = re.sub(r"\=", " = ", text)
    text = re.sub(r"'", " ", text)
    text = re.sub(r"(\d+)(k)", r"\g<1>000", text)
    text = re.sub(r":", " : ", text)
    text = re.sub(r" e g ", " eg ", text)
    text = re.sub(r" b g ", " bg ", text)
    text = re.sub(r" u s ", " american ", text)
    text = re.sub(r"\0s", "0", text)
    text = re.sub(r" 9 11 ", "911", text)
    text = re.sub(r"e - mail", "email", text)
    text = re.sub(r"j k", "jk", text)
    text = re.sub(r"\s{2,}", " ", text)

    text = text.lower().split()

    if remove_stopwords:
        text = [w for w in text if not w in stops]

    if w2v is not None:
        text = ['#hashtag' if ((w not in w2v) and '#' in w) else w for w in text]
```

Подготовка данных

```
word2vec_path = 'word_embeddings/word2vec/word2vec_twitter_model/word2vec_twitter_model.bin'
word2vec = w2v_reader.Word2Vec.load_word2vec_format(word2vec_path,
                                                    binary=True)
```

```
main_emotions = ['anger','fear','joy','sadness']

full_data={}
full_Y = {}
for emotion in main_emotions:
    full_data[emotion] = get_emotion_data(emotion)
    for data in full_data[emotion]:
        data['cleaned_text'] = data['text'].map(lambda s: text_to_wordlist(s, w2v = word2vec))
    full_Y[emotion] = [data['intensity'] for data in full_data[emotion]]
```

```
MAX_SEQUENCE_LENGTH = 30
tokenizer = Tokenizer(filters='"$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n')
tokenizer.fit_on_texts(np.hstack([data['cleaned_text'] for data
                                  in full_data['anger']+full_data['fear']+full_data['joy']+full_data['sadness']]))
```

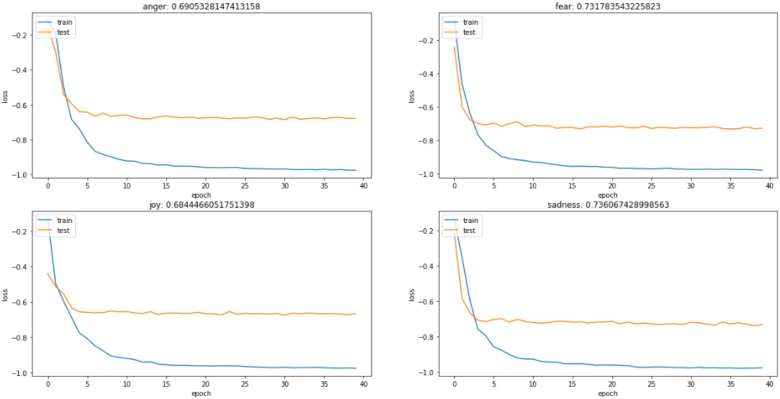
```
X_sequences = {}
for emotion in main_emotions:
    X_sequences[emotion] = [pad_sequences(tokenizer.texts_to_sequences(data['cleaned_text'].values),
                                          maxlen=MAX_SEQUENCE_LENGTH) for data in full_data[emotion]]
```

```
full_data['anger'][0].head(3)
```

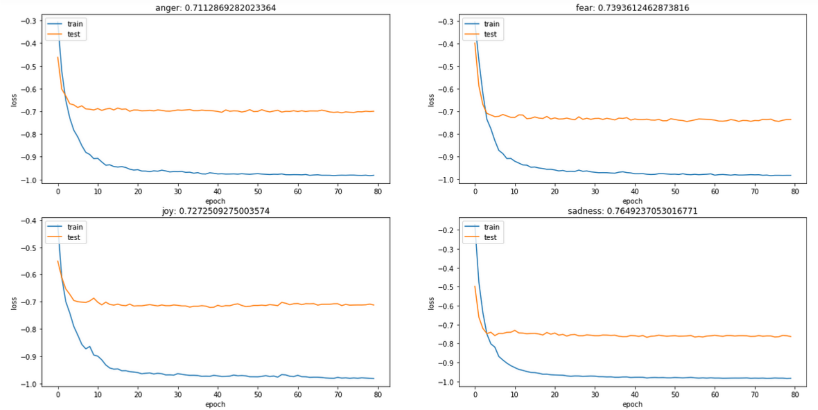
	id	text	emotion	intensity	cleaned_text
0	10000	How the fu*k! Who the heck! moved my fridge!... should I knock the landlord door. #angry #mad ##	anger	0.938	how the fu k ! who the heck ! moved my fridge ! should i knock the landlord door #angry #mad ##
1	10001	So my Indian Uber driver just called someone the N word. If I wasn't in a moving vehicle I'd have jumped out #disgusted	anger	0.896	so my indian uber driver just called someone the n word if i was not in a moving vehicle i would have jumped out #disgusted
2	10002	@DPD_UK I asked for my parcel to be delivered to a pick up store not my address #fuming #poorcustomerservice	anger	0.896	username i asked for my parcel to be delivered to a pick up store not my address #fuming #poorcustomerservice

```
def pearson_loss(y_true, y_pred):
    numerator = -K.sum((y_true-K.mean(y_true))*(y_pred-K.mean(y_pred)))
    denominator = ( K.sqrt(K.sum((K.square(y_true-K.mean(y_true))))) * K.sqrt(K.sum((K.square(y_pred-K.mean(y_pred)))))\
                   +K.epsilon() )
    return numerator/denominator


def get_model(embedding_matrix):
    model = Sequential()
    model.add(Embedding(len(tokenizer.word_index)+1, EMBEDDING_DIM, weights = [embedding_matrix],trainable=False))
    model.add(Conv1D(filters=200, kernel_size=3, padding='same', activation='relu'))
    model.add(MaxPooling1D(pool_size=2))
    model.add(Dropout(0.3))
    model.add(Bidirectional(LSTM(150, activation='relu', dropout=0.2, kernel_initializer='he_normal',
                                 return_sequences=True)))
    model.add(Bidirectional(LSTM(80, dropout=0.2,kernel_initializer='he_normal')))
    model.add(Dense(50, activation='relu',kernel_initializer='he_normal'))
    model.add(Dropout(0.3))
    model.add(Dense(1, activation='sigmoid'))
    model.compile(loss=pearson_loss, optimizer="adam")

    return model
```

Обучим вышеуказанную модель. Используем twitter word2vec (dim=400) для слоя эмбедингов.



Добавим к word2vec эмбедингу дополнительную координату, в которую запишем значение полученное из расширенного AFINN лексикона



NN+GB

```
gb_prediction={}
lstm_prediction={}
scores = {}
for emotion in main_emotions:
    gb_prediction[emotion] = pickle.load(open( "features/HE_AI_AFINN_GLOVE_TFIDF_gb_{emotion}.p".format(emotion=emotion),
                                              "rb" ))
    lstm_prediction[emotion] = afinn_models[emotion]['model'].predict(X_sequences[emotion][2]).reshape(-1)
    scores[emotion]=evaluate(full_Y[emotion][2],np.mean([gb_prediction[emotion],lstm_prediction[emotion]],axis=0))
    print('{0} test score: {1}'.format(emotion,scores[emotion]))

np.mean([scores[emotion] for emotion in main_emotions],axis=0)
```

```
anger test score: (0.72964462630772731, 0.70998318261230065, 0.56628688240470315, 0.53613200563175067)
fear test score: (0.75744605983811619, 0.74126968473640642, 0.61652872152914739, 0.58207835501853522)
joy test score: (0.74059459126028815, 0.73980656759718677, 0.50247381392713786, 0.50016711888857357)
sadness test score: (0.76826862500849002, 0.76405787146964421, 0.58103474289037027, 0.55539429741776825)

array([ 0.74898848,  0.73877933,  0.56658104,  0.54344294])
```

```
gb_prediction={}
lstm_prediction={}
scores = {}
for emotion in main_emotions:
    gb_prediction[emotion] = pickle.load(open( "features/HE_AI_AFINN_GLOVE_TFIDF_EXTENDED_gb_{emotion}.p"\
                                              .format(emotion=emotion),
                                              "rb" ))
    lstm_prediction[emotion] = afinn_models[emotion]['model'].predict(X_sequences[emotion][2]).reshape(-1)
    scores[emotion]=evaluate(full_Y[emotion][2],np.mean([gb_prediction[emotion],lstm_prediction[emotion]],axis=0))
    print('{0} test score: {1}'.format(emotion,scores[emotion]))

np.mean([scores[emotion] for emotion in main_emotions],axis=0)
```

```
anger test score: (0.74013314191334922, 0.72199095398980051, 0.57610071640811367, 0.54763195236341877)
fear test score: (0.76014011404321657, 0.74426693383446441, 0.61917539268805228, 0.58478102796238884)
joy test score: (0.74371867987023488, 0.74525594606886325, 0.50660556967851056, 0.506111432232213)
sadness test score: (0.77318133020875113, 0.76911058182431202, 0.56986110817270563, 0.54774100790718994)

array([ 0.75429332,  0.7451561 ,  0.5679357 ,  0.54656636])
```

Team Name	r avg. (rank)	r fear (rank)	r joy (rank)	r sadness (rank)	r anger (rank)
w2v_gb_ext	0.7542	0.7601	0.7437	0.7731	0.7401
w2v_gb	0.7489	0.7574	0.7405	0.7682	0.7296
1. Prayas	0.747 (1)	0.732 (1)	0.762 (1)	0.732 (1)	0.765 (2)
w2v_afinn	0.7357	0.7393	0.7272	0.7649	0.7112
2. IMS	0.722 (2)	0.705 (2)	0.726 (2)	0.690 (4)	0.767 (1)
3. SeerNet	0.708 (3)	0.676 (4)	0.698 (6)	0.715 (2)	0.745 (3)
4. UWaterloo	0.685 (4)	0.643 (8)	0.699 (5)	0.693 (3)	0.703 (7)
5. IITP	0.682 (5)	0.649 (7)	0.713 (4)	0.657 (7)	0.709 (5)
6. YZU NLP	0.677 (6)	0.666 (5)	0.677 (8)	0.658 (6)	0.709 (5)
7. YNU-HPCC	0.671 (7)	0.661 (6)	0.697 (7)	0.599 (9)	0.729 (4)
8. TextMining	0.649 (8)	0.604 (10)	0.663 (9)	0.660 (5)	0.668 (10)
9. XRCE	0.638 (9)	0.629 (9)	0.657 (10)	0.594 (10)	0.672 (9)
10. LIPN	0.619 (10)	0.58 (11)	0.639 (11)	0.583 (11)	0.676 (8)
11. DMGroup	0.571 (11)	0.55 (12)	0.576 (12)	0.556 (12)	0.603 (11)
12. Code Wizards	0.527 (12)	0.465 (16)	0.534 (15)	0.532 (14)	0.578 (13)
13. Todai	0.522 (13)	0.470 (15)	0.561 (13)	0.537 (13)	0.520 (16)
14. SGNLP	0.494 (14)	0.486 (14)	0.512 (16)	0.429 (18)	0.550 (14)
15. NUIG	0.494 (14)	0.680 (3)	0.717 (3)	0.625 (8)	-0.047 (21)
16. PLN PUCRS	0.483 (16)	0.508 (13)	0.460 (19)	0.425 (19)	0.541 (15)
17. H.Niemtsov	0.468 (17)	0.412 (17)	0.511 (17)	0.437 (17)	0.513 (17)
18. Tecnolengua	0.442 (18)	0.373 (18)	0.488 (18)	0.439 (16)	0.469 (18)
19. GradAscent	0.426 (19)	0.356 (19)	0.543 (14)	0.226 (20)	0.579 (12)
20. SHEF/CNN	0.291 (20)	0.277 (20)	0.109 (20)	0.517 (15)	0.259 (19)
21. deepCybErNet	0.076 (21)	0.176 (21)	0.023 (21)	-0.019 (21)	0.124 (20)
Late submission					
∗ SiTAKA	0.631	0.626	0.619	0.593	0.685

	Team																					
Features	1	2	3	4	5	6	7	8	9	*	10	11	12	13	14	15	16	17	18	19	20	21
N-grams				✓									✓									
CN													✓									
WN				✓									✓			✓						
Word Embeddings	✓	✓	✓	✓	✓	✓	✓	✓		✓			✓	✓	✓	✓				✓		
Glove			✓	✓	✓	✓	✓	✓		✓				✓		✓				✓		
Emoji Vectors			✓	✓																		
Word2Vec	✓	✓	✓	✓																		
Other								✓					✓		✓							
Sentence Embeddings																						
CNN	✓	✓				✓	✓	✓		✓					✓						✓	✓
LSTM	✓	✓			✓	✓	✓	✓						✓		✓				✓		
Other				✓												✓				✓	✓	
Affective Lexicons		✓	✓	✓	✓	✓		✓	✓	✓				✓				✓	✓	✓		
AFINN	✓	✓	✓		✓			✓														
ANEW		✓																				
BingLiu	✓	✓	✓		✓			✓	✓													
Happy Ratings		✓																				
Lingmotif																			✓			
LIWC																	✓					
MPQA	✓	✓	✓		✓			✓														
NRC-Aff-Int	✓		✓	✓				✓														
NRC-EmoLex	✓	✓	✓	✓	✓			✓	✓													
NRC-Emoticon-Lex	✓		✓	✓				✓					✓									
NRC-Hash-Emo	✓	✓	✓	✓	✓			✓	✓													
NRC-Hash-Sent		✓	✓	✓	✓			✓														
NRC-Hashtag-Sent.	✓		✓	✓																		
NRC10E	✓	✓	✓					✓														
Sentiment140	✓	✓	✓	✓				✓														
SentiStrength		✓	✓					✓														
SentiWordNet	✓	✓	✓	✓	✓			✓														
Vader					✓																	
Word.Affect			✓																			
In-house lexicon	✓								✓								✓					
Linguistic Features									✓													
Dependency Parser									✓														