Detection and Characterization of Stance on Social Media

Part 2
Detection and Characterization of Stance on Social Media

Abeer Aldayel, School of Informatics, University of Edinburgh
Kareem Darwish, Qatar Computing Research Institute, HBKU
Walid Magdy, School of Informatics, University of Edinburgh
Part2: **Stance modeling in social media**

2.1 Stance detection modeling

2.2 Most effective classification features.

2.3 SOTA Stance detection methods
   (semi-supervised, and unsupervised)
Supervised Stance Detection
Supervised SD

• Train a text based classifier on Tweets:
  • Case study: Egyptian 2013 Coup (pro-coup/anti-coup)
    • Period of study: June 21 – Oct. 1, 2013
    • Tweet level classification
  • Features: word unigrams, word bigrams, hashtags
  • Labeled data: 1,000 tweets – pro/anti/neutral
  • Evaluation: 20-fold cross validation
  • Avg accuracy: 87%

Supervised SD

• Train a text based classifier on Tweets:
  • Case study: Egyptian 2013 Coup (pro-coup/anti-coup)
  • Given that classification was done on tweets, we can observe changes in stance.
    • June-21: We will continue to revolt till we reach freedom. Gathering revolution from Alexandria to Cairo to oust Morsi, the sheep.
    • July-19: The Mohandseen march is closing the main streets till the police station #NoToMilitaryCoup
  • Percentage of change 2-3%

Supervised SD

• Train a text based classifier on users:
  • Case study: Support for ISIS (ISIS vs. Islamic State)
    • User level classification
    • Features: word unigrams, hashtags, user mentions
    • Labeled data: > 14,000 users – pro/anti
    • Evaluation: 10-fold cross validation

<table>
<thead>
<tr>
<th></th>
<th>Precision</th>
<th>Recall</th>
<th>F1-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pro-ISIS</td>
<td>89.6</td>
<td>83.7</td>
<td>86.6</td>
</tr>
<tr>
<td>Anti-ISIS</td>
<td>84.7</td>
<td>90.3</td>
<td>87.4</td>
</tr>
<tr>
<td>Average</td>
<td>87.2</td>
<td>87.0</td>
<td>87.0</td>
</tr>
</tbody>
</table>

• We can predict who will end up supporting ISIS later with 87% accuracy

Supervised SD

• Train a text based classifier on users:
  • Case study: Islamophobia in the US (pro/anti)
    • ISIS carries out terrorist attacks in Paris 11/2015
  • User level classification
  • Features: word unigrams/hashtags/user mentions/RT
  • Labeled data: 1,534 user tweets – pro/anti ➔ 44k users
    • I feel horrible that people who practice Islam have to apologize for the #ParisAttack - Muslim people aren't responsible; terrorists are.
    • Why are muslims even allowed out of their garbage countries? We need to take out the trash #KillAllMuslims #DeportAllMuslims #RemoveKebab

Supervised SD

• Train a text based classifier on users:
  • Case study: Islamophobia in the US (pro/anti)
  • Can we predict who will have Islamophobic views?
  • Evaluation: 200 tweets before incident for training

<table>
<thead>
<tr>
<th></th>
<th>w/ prior views</th>
<th>w/o prior views</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Hashtags</td>
<td>RT</td>
</tr>
<tr>
<td>Positive Prec.</td>
<td>84</td>
<td>89</td>
</tr>
<tr>
<td>Negative Prec.</td>
<td>75</td>
<td>83</td>
</tr>
</tbody>
</table>

Supervised SD

• Map users into latent space prior to classification:
  • Pick a set of “exemplar users”, and use similarity to them as features

Path 1:

\[ p_1(U_i|U_j) = p(IE_l|U_j) p(U_i|IE_l) = \frac{11}{43} = \frac{1}{12} \]

Path 2:

\[ p_2(U_i|U_j) = p(IE_k|U_j) p(U_i|IE_k) = \frac{1}{4} \cdot \frac{1}{100} = \frac{1}{400} \]

Combining Paths:

\[ \sum_p p_n(U_i|U_j) = \frac{1}{12} + \frac{1}{400} = 0.0858 \]

Or:

\[ 1 - \prod_p (1 - p_n(U_i|U_j)) = 0.0856 \]

P that all paths are incorrect
Supervised SD

• Map users into latent space prior to classification:
  • Pick a set of “exemplar users”, and use similarity to them as features
    • Case study: Islamophobia dataset
      • Computed similarity based on: RT/Hashtags
      • Size of latent space: 100 users
      • Training set: 100 users, Test set: 2,607 users
  • Compared raw features vs. using the features to compute similarity

<table>
<thead>
<tr>
<th>SVM Classifier</th>
<th>RT</th>
<th>HASH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Measure</td>
<td>SIM</td>
<td>Raw</td>
</tr>
<tr>
<td>POS F1</td>
<td>0.76</td>
<td>0.69</td>
</tr>
<tr>
<td>NEG F1</td>
<td>0.92</td>
<td>0.90</td>
</tr>
<tr>
<td>Macro-F1</td>
<td>0.84</td>
<td>0.80</td>
</tr>
</tbody>
</table>

Supervised SD

• Advantages:
  • Simple

• Disadvantages:
  • Accuracy seems to be capped
  • Requires training data

• Observations:
  • Works even with non-topical content
  • Network interactions (RT) better than actual content (hashtags)
  • People’s views are durable
  • User similarity can improve SD
  • May be can learn something from social psychology
Semi-Supervised & Unsupervised Stance Detection
To be Social is to be Human!

YOUR ENGINEERING EXPERIENCE LOOKS GREAT, BUT YOUR SOCIAL MEDIA SCORE IS NEARLY ZERO.

YOU HAVE NO FRIENDS, NO FOLLOWERS, AND NO SOCIAL INFLUENCE WHATSOEVER.

BECAUSE I FOCUS ON MY WORK!

NO, I'M PRETTY SURE YOU'RE DEAD.
Humans are Homophilous & They Exert Social Pressure
Semi-Supervised SD

• Fundamental assumptions:
  • Users have strong homophily and form echo chambers.
  • Users rarely change their positions

• Observations:
  • Users RT much more than they tweet
  • Tweeting frequency resembles a zipf distribution

• Thus:
  • Users who retweet the same tweets have identical views!
  • If we tag most active users, then we can propagate the labels
Semi-Supervised SD

• Label propagation
Semi-Supervised SD

• Some shared tweets between @jtstover & @popy_panayotou

Donald Trump Jr.
@DonaldJTrumpJr

Jim, did you or your colleagues take any responsibility for your rhetoric in constantly calling my family Racists and Nazis for 2 years when my wife actually opened an exploding envelope of white powder intended for me? What about when it happend to my brother?

Donald J. Trump
@realDonaldTrump

…I agree with their stance 100%, and the United States is likewise taking a very hard line on illegal immigration. The Prime Minister is working very hard on the economy of Italy - he will be successful!

Jim Acosta
@Acosta · Oct 25, 2018

Trump talked about the bombs sent to former public officials at his rally in Wisconsin. But once again he didn’t mention CNN. On top of all that he criticized the “media” for coverage he doesn’t like. And, of course, he took no responsibility for his own rhetoric.

2:58 PM · Oct 25, 2018 · Twitter for iPhone

26.4K Retweets  64.5K Likes
Semi-Supervised SD

• Procedure:
  • Given a set of labeled users, propagate their tags to all their tweets
  • For each unlabeled user, count all their tweets that have tags per label
    • If all tagged tweets belong to one label & # of tagged tweets > threshold (ex. min. 5) → propagate label to user
      • Ex. 6 pro tweets, 0 anti tweets ✓
      • Ex. 10 pro tweets, 1 anti tweets ❌
  • Else do nothing
  • Repeat until no more users can be tagged
def labelPropagationTweets(labelFile, tweetsFile, threshold):
    # load training set (initial set of labeled users)
    userLabels = defaultdict()
    with open(labelFile) as f:
        for line in f:
            parts = line.strip().lower().split('\t')
            if len(parts) >= 2:
                userLabels[parts[0]] = parts[1]

    # load tweets of labeled users and assign user labels to tweets
    # if tweet is mentioned by different groups, it gets a tag of 'UNK' and is later ignored
    tweetLabels = defaultdict()
    with open(tweetsFile) as f:
        for line in f:
            parts = line.strip().lower().split('\t')
            if len(parts) >= 2:
                user = parts[0]
                tweet = cleanTweet(parts[1])
                if user in userLabels:
                    if tweet not in tweetLabels:
                        tweetLabels[tweet] = userLabels[user]
                    elif tweetLabels[tweet] != userLabels[user]:
                        tweetLabels[tweet] = 'UNK'
# iterate over tweets of all unlabeled users, and count the number of tweets
# they have retweeteded from different groups
newUserLabels = defaultdict()
with open(tweetsFile) as f:
    for line in f:
        parts = line.strip().lower().split('t')
        if len(parts) >= 2:
            user = parts[0]
            tweet = cleanTweet(parts[1])
            if tweet in tweetLabels and tweetLabels[tweet] != 'UNK':
                if user not in newUserLabels:
                    newUserLabels[user] = dict()
                if tweetLabels[tweet] not in newUserLabels[user]:
                    newUserLabels[user][tweetLabels[tweet]] = 1
                else:
                    newUserLabels[user][tweetLabels[tweet]] += 1

# if users have tweets with single labels that are more than the threshold
# then add to the final list
finalUserLabels = defaultdict()
for user in newUserLabels:
    if len(newUserLabels[user]) == 1:
        for u in newUserLabels[user]:
            if newUserLabels[user][u] > threshold:
                finalUserLabels[user] = u

# put back the training set that we started with
for user in userLabels:
    finalUserLabels[user] = userLabels[user]
return finalUserLabels
## Semi-Supervised SD – Trump Dataset (11,587 users)

<table>
<thead>
<tr>
<th>Iteration</th>
<th># of Labeled Users</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
<td>100.0%</td>
</tr>
<tr>
<td>1</td>
<td>3,246</td>
<td>98.9%</td>
</tr>
<tr>
<td>2</td>
<td>4,988</td>
<td>99.2%</td>
</tr>
<tr>
<td>3</td>
<td>5,055</td>
<td>99.1%</td>
</tr>
<tr>
<td>4</td>
<td>5,057</td>
<td>99.1%</td>
</tr>
<tr>
<td>5</td>
<td>5,057</td>
<td>99.1%</td>
</tr>
</tbody>
</table>

- **Precautions:**
  - If labels are contaminated, disaster strikes
  - Recall is low
SD – Label Propagation

• Case study: Kavanaugh Nomination to Supreme Court (pro/anti)
  • Narrowest successful nomination (50-48) since 1881
  • Data collection: 23M tweets from 687K users
  • Labeled data: 41 users (29 pro/12 anti)
  • Label propagation results: 66K users (27K pro/39K anti)
  • Estimated labeling accuracy: > 98%
  • Used RT as feature with fastText classifier with high threshold (>0.90) to label more users:
    • Labeled users: 128K users (57K pro/71K anti)
    • Estimated labeling accuracy: 96% (on a sample of 100)

SD – Label Propagation

- Case study: 2018 Turkish elections (devam/tamam)
  - Data collection: 108M tweets (April 29 – June 23) from 687K users
  - Labeled data: 3,866 users who explicitly specify affiliation

- Label propagation results: 652K users
- Estimated labeling accuracy: > 95% (based on 200 user sample)


SD – Semi-Supervised Learning

• Advantages:
  • Simple
  • Very high accuracy

• Disadvantages:
  • Requires training data
  • Identifies users with strong opinions

• Observations:
  • Similarity between users is a strong feature
Manual annotation is dreadful
Unsupervised SD

- Motivation: Similar users ➔ similar stances
- But clustering in high-dimensional spaces does not work!
Unsupervised SD – Method 1

- Datasets: 5 sampled sets from Kavanaugh dataset (23M tweets), Turkish election dataset (108M), and Trump dataset (4M tweets) – Total 15 sets
- Dimensionality reduction: Force directed graph, t-SNE, UMAP
- Clustering: DBSCAN, Mean Shift
- Sample sizes: 50K, 100K, 250K, 1M
- Features to compute similarity: RT, hashtags, tweets
- Top (most active) users to cluster: 500, 1,000, 5,000

Unsupervised SD – Method 1

• Results:
  • Purity > 80% & > 10% of users were clustered.

• Best setup:
  • Features: RT
  • Tweet set size:
    • 100K tweets ➔ purity > 90% || 250K tweets ➔ purity > 98%
  • No. of users to cluster: > 500
  • Dimensionality reduction: UMAP
  • Clustering: mean shift

<table>
<thead>
<tr>
<th># of Users</th>
<th>Feature(s)</th>
<th>Dim Reduce</th>
<th>Peak Detect</th>
<th>Avg. Purity</th>
<th>Avg. # of Clusters</th>
<th>Avg. Cluster Size</th>
<th>Avg. Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>R</td>
<td>FD</td>
<td>Mean Shift</td>
<td>98.7</td>
<td>2.5</td>
<td>171.3</td>
<td>68.6</td>
</tr>
<tr>
<td>500</td>
<td>R</td>
<td>UMAP</td>
<td>Mean Shift</td>
<td>98.5</td>
<td>2.1</td>
<td>179.9</td>
<td>72.0</td>
</tr>
</tbody>
</table>

# load user tweets

df_text = pd.read_csv(args.source_file, header=None, usecols=[0, 1], sep='\t', error_bad_lines=False)
df_text.columns = ['User', 'Text']
df_text = df_text.apply(lambda s: s.str.strip())
df_text.loc[:, 'User'] = df_text.User.str.lower()

# Extract retweeted accounts and work only with those rows

df_text['Retweet'] = df_text.Text.apply(extract_rt)
df_text.dropna(subset=['Retweet'], inplace=True)

# Sample users -- take `sample_size` most active

min_nunique_retweet = 5
sample_size = 5000

users_of_interest = df_text.groupby('User')['Retweet'].nunique().where(lambda x: x >= min_nunique_retweet).dropna().sort_values(ascending=False).head(n=sample_size)

# Filter df_text to have only the Tweets of the sampled users

df_text = df_text[df_text.User.isin(users_of_interest.index)]
# Calculate similarity
user_feature_counts = df_text.groupby(['User', 'Retweet']).size()
user2idx = Enumerator()
feature2idx = Enumerator()

row_ind = []
col_ind = []
data = []
for (user, feature), count in user_feature_counts.items():
    row_i = user2idx[user]
    col_i = feature2idx[feature]
    row_ind.append(row_i)
    col_ind.append(col_i)
    data.append(count)

user_feature_matrix = sparse.csr_matrix((data, (row_ind, col_ind)))
user2user_sim = cosine_similarity(user_feature_matrix).clip(max=1.0)

# Dimensionality reduction
user_points = UMAP(metric='precomputed').fit_transform(1 - user2user_sim)  # works with distances, NOT similarity

# Scale user vectors between -1 and 1
scaler = MinMaxScaler(feature_range=(-1, 1))
user_points_scaled = scaler.fit_transform(user_points)
Unsupervised SD

- Implementation

```
idx2user = {v:k for k, v in user2idx.dict.items()}
df_user = pd.DataFrame(user_points_scaled, index=map(idx2user.get, range(len(user_points_scaled))))

# Clustering
clusters = MeanShift(cluster_all=False, bin_seeding=True).fit(df_user.values)
df_user['cluster_id'] = clusters.labels_

# Regard users in very small clusters as unclustered
for c in df_user['cluster_id'].unique():
    if c == -1:
        continue
    if len(df_user[df_user['cluster_id'] == c]) < len(df_user[df_user['cluster_id'] != -1]) * 0.01:
        df_user.loc[df_user['cluster_id'] == c, 'cluster_id'] = -1

# Generate output
df_user[df_user.cluster_id != -1].to_csv(args.output_file, header=False)
```
Unsupervised SD – Trump Dataset (11,587 users)

<table>
<thead>
<tr>
<th># of Labeled Users</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>7,042</td>
<td>94.7%</td>
</tr>
</tbody>
</table>

• Precautions:
  • Only tags active users
  • Recall is relatively low
Caution:

MUST USE umap-learn v. 0.3.6

v.0.4.4 DOES NOT WORK
Unsupervised SD – Method 1 – Extension

- What if we only have a few topical tweets from a user?
  - Answer: get their timeline tweets, and then use method

<table>
<thead>
<tr>
<th>Topic</th>
<th>Unsupervised on timeline expansion</th>
<th>SVM$_{RT}$</th>
<th>BERT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kavanaugh</td>
<td>$A$ 84.6</td>
<td>$P$ 84.2</td>
<td>$R$ 84.2</td>
</tr>
<tr>
<td>Vaccine</td>
<td>$A$ 96.3</td>
<td>$P$ 83.3</td>
<td>$99.0$</td>
</tr>
<tr>
<td>Ilhan</td>
<td>$A$ 91.9</td>
<td>$P$ 91.6</td>
<td>$R$ 92.1</td>
</tr>
<tr>
<td>Gun Control</td>
<td>$A$ 95.8</td>
<td>$P$ 90.6</td>
<td>$R$ 86.4</td>
</tr>
<tr>
<td>Police Racism</td>
<td>$A$ 100.0</td>
<td>$P$ 100.0</td>
<td>$R$ 100.0</td>
</tr>
<tr>
<td>Climate Change</td>
<td>$A$ 96.2</td>
<td>$P$ 97.8</td>
<td>$R$ 88.9</td>
</tr>
<tr>
<td>Midterm</td>
<td>$A$ 100.0</td>
<td>$P$ 100.0</td>
<td>$R$ 100.0</td>
</tr>
<tr>
<td>Immigration</td>
<td>$A$ 100.0</td>
<td>$P$ 100.0</td>
<td>$R$ 100.0</td>
</tr>
<tr>
<td>Average</td>
<td>$A$ 95.6</td>
<td>$P$ 93.4</td>
<td>$R$ 93.8</td>
</tr>
</tbody>
</table>

No expansion: 78.9  80.4  68.7  66.6

Unsupervised SD – Method 2

• What if we don’t have retweets?
• We can use embeddings-based clustering:
  • Given a tweet ➔ embeddings vector (Universal Sentence Encoder)
  • Take average of tweet embedding vectors to represent user

Project: UMAP

Cluster: Mean shift


Unsupervised SD – Method 2

• Turkish election dataset:
  • Users: 5,960
  • Tweets: 1.48M
• Universal sentence encoder (CNN)

<table>
<thead>
<tr>
<th>Supporters</th>
<th>Users</th>
<th>Tweets</th>
</tr>
</thead>
<tbody>
<tr>
<td>pro-Erdoğan</td>
<td>1,772</td>
<td>561,510</td>
</tr>
<tr>
<td>anti-Erdoğan w/o party affiliation</td>
<td>2,115</td>
<td>516,166</td>
</tr>
<tr>
<td>pro-CHP</td>
<td>29</td>
<td>171,201</td>
</tr>
<tr>
<td>pro-IYI</td>
<td>890</td>
<td>168,442</td>
</tr>
<tr>
<td>pro-HDP</td>
<td>354</td>
<td>61,274</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>5,960</td>
<td>1,478,593</td>
</tr>
</tbody>
</table>

Unsupervised SD – Method 2

• Pro’s:
  • Works without network features
  • Can produce fine-grained stance detection
  • CNN-based is fast

• Con’s:
  • Transformer-based is slow

def clusterUsers(df, embed: Callable, min_tweets=3, user_col="username", tweet_col="norm_tweet", save_at="temp.npz", min_dist=0.0, n_neighbors=90, **kwargs):

    # load and group users
    # map to embeddings space and take average
    gs = df.groupby(user_col)
    users = list()
vectors = list()
for user, frame in tqdm(gs):
    if len(frame) < min_tweets:
        continue
    try:
        tweets = frame[tweet_col]
        vec = np.mean(np.array(embed(tweets.tolist())), axis=0)
        users.append(user)
vectors.append(vec)
    except Exception as e:
        print(f"ERROR at:{user}")
        print(e)
        print()

users: np.ndarray = users
vectors: np.ndarray = vectors
# project to lower dimensional space
standard_embeddings = UMAP(
    random_state=42,
    n_components=2,
    n_neighbors=n_neighbors,
    min_dist=min_dist,
    metric='cosine', **kwargs
).fit_transform(vectors)
print("Projection complete")

params = dict()

# cluster using HDBSCAN
clusterer = cluster_embeddings(standard_embeddings, **kwargs)
params['clusters'] = clusterer.labels_
params['allow_pickle'] = True
np.savez(open(save_at + '.cluster', 'wb'), users=np.array(users),
         vectors=np.array(vectors), umap=np.array(standard_embeddings),
         clusters=np.array(clusterer.labels_))

outputFile = open(save_at + '.clusters.txt', mode='w')
for i in range(len(clusterer.labels_)):
    outputFile.write(str(users[i]) + '	' + str(clusterer.labels_[i]) + '
')
outputFile.close()
Unsupervised SD – Trump Dataset (11,587 users)

<table>
<thead>
<tr>
<th># of Labeled Users</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>9,995</td>
<td>88.7%</td>
</tr>
</tbody>
</table>

- Precautions:
  - Tags most users
  - Accuracy is slightly lower
Unsupervised SD

• Advantages:
  • Simple
  • Very high accuracy
  • No training data required

• Disadvantages:
  • Identifies users with strong opinions
  • Only works on topics with strong polarization
  • Clusters still need to be labeled

• Observations:
  • We learn the general leaning of popular news sites and Twitter accounts across multiple topics
  • We can extend to measure polarization
Not So Fast

Stance Detection Done

NO Manual Annotation & Super Accurate
What is Best Course for SD?

• Think through your application

• Start with an unsupervised method to tag vocal users:
  • If network features are present, use retweets as features
  • If they are not present, use embeddings

• Fallback to other methods to tag less vocal users:
  • Expand user tweets with timeline tweets and cluster
  • Use supervised learning:
    • SVM & FastText are fast
    • BERT is more accurate but needs fine-tuning (expensive)

• Look for clues in user profiles:
  • #Resist & #VoteBlue vs. #MAGA & #KAG2020
Conclusion

unsupervised first
Make sure that topic is polarizing

supervised second
Spot check results to estimate errors

Happy stance detection ☺.
End of part 2

…To be continued with part3