Discovering the Hierarchy of Social Ideas from the Contributions of Nonprofits

Yuxiao Sun

I. **Introduction**

Our society is built upon a variety of shared beliefs: environment protection, human rights, racial inequality, gender inequality, just to name a few. Revealing the hierarchy of these shared ideas, however, is more difficult than it appears. For one thing, different individuals would almost certainly come up with different rankings; for another, even with mass surveys, there is still an important discrepancy between what people say and what people do.

This paper takes an empirical approach to the above inquiry. In particular, I view the operations of nonprofits as a natural experiment of our society fulfilling part of its value system. Each nonprofit, as a venture of ideas summarized by its mission, receives contributions from individuals, foundations and government as an exchange of its agreed-upon level of services provision. Thus by collecting data on received contributions of different nonprofits, we can trace back the hierarchy of social services, and eventually social ideas attracting these contributions.

State it mathematically:

(1) Received contributions = F1 (Service) *Classification Models: DT, KNN, SVM, RF, etc.*

(2) Service = F2 (Mission, NTEE Type)

(3) Mission = F3 ($\sum Pr \times Idea(Topic)$ *)* *Topic Modeling: LDA with various parameters*

(4) Idea (Topic) = F4 ($\sum Pr \times Word$ *)*

The data I use is from IRS Form 990 annual extract 2008-2013. Each year, about 250,000 nonprofits are eligible to file Form 990 Tax exempt form[1]. For each organization I am averaging the received contributions within the five year to avoid yearly fluctuation. The services of nonprofits are explained by two important variables: NTEE type and mission statements.

The National Taxonomy of Exempt Entities (NTEE) classification system was developed by the National Center for Charitable Statistics to classify nonprofit organizations. It has 26 major categories:

A - Arts, Culture & Humanities	N - Recreation & Sports
B – Education	O - Youth Development
C - Environment	P - Human Services
D – Animal-Related	Q - International, Foreign Affairs & National Security
E - Health Care	R - Civil Rights, Social Action & Advocacy
F - Mental Health & Crisis Intervention	S - Community Improvement & Capacity Building
G - Voluntary Health Associations & Medical Disciplines	T - Philanthropy, Voluntarism & Grantmaking Foundations
H - Medical Research	U - Science & Technology
I - Crime & Legal-Related	V - Social Science
J - Employment	W - Public & Societal Benefit
K - Food, Agriculture & Nutrition	X - Religion-Related
L - Housing & Shelter	Y - Mutual & Membership Benefit
M - Public Safety, Disaster Preparedness & Relief	Z – Unknown

The NTEE classification is “official” and exclusive, yet it is not without pitfalls. Firstly, as a self-report category, there are about 20%-30% nonprofits are classified Z-Unknown (figure 1). What's more, most of the NTEE categories are too broad to capture concise social ideas. Thus the NTEE variable is mainly used as a control variable/fixed effect variable to explain variations in nonprofits' services. And the social ideas are captured instead by nonprofits' mission statements, which are usually a long sentence

1 They are usually the larger nonprofits with receipts>200,000 and assets>500,000, see here for definition https://www.irs.gov/charities-non-profits/form-990-series-which-forms-do-exempt-organizations-file-filing-phase-in

or a short paragraph discussing the objectives and operations of nonprofits. By exploiting content analysis techniques such as topic modeling we can generate “topic groups” that might be a better correspondence of social ideas behind the nonprofits.

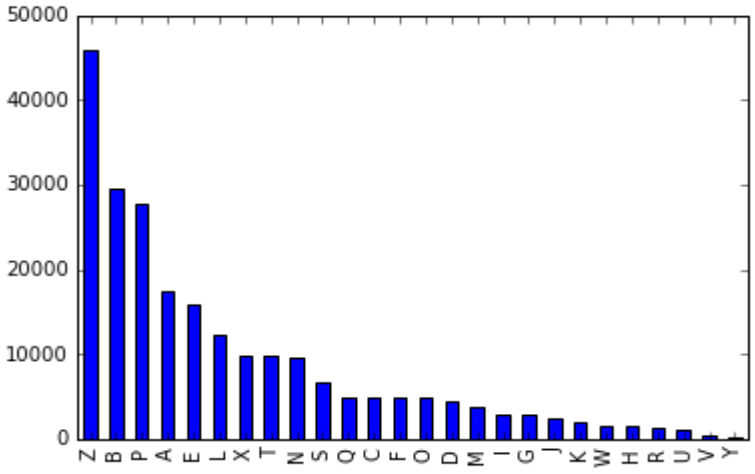


Figure 1 Distribution of NTEE types

To be more specific, I am going to use the LDA model to extract features from the mission statements. As a technique of topic modeling, in LDA each document is viewed as a probability representation of all the topics and each topic is a probability distribution over the word space. Thus if we view each topic as a relatively independent social idea then the LDA produces a probability distribution of each mission statement over a high dimensional space of social ideas. The benefits of using LDA for feature extraction rather then a simple documents-words transformation is two-fold. Firstly, it greatly reduces the dimension and sparsity of the feature matrix without losing too much of the interpretability; Secondly, compare to a single word, a group of words (i.e., a topic) might be a better representation of a social idea/belief.

After generating the LDA model for the mission statements (function (3) and (4)), we can use the documents-topics matrix and NTEE variables as features to explain variations in service types

(function (2)) and finally contributions (function (1)). The fitness of the classification models reflect the feasibility of deriving hierarchy of social ideas from nonprofits' received contributions; whereas the parameters of the classification models give us clues on the hierarchy itself.

II. Work-flow

The model development work-flow is described as follows:

1. $ Python3 merge.py: loading and preprocessing data

1) Merge different datasets;

2) Averaging contributions over the five years;

3) Remove organizations with missing values (240,000+ to 207,000+).

```
import pandas as pd
df1=pd.read_table('bmf.txt', header=None,names=['EIN','NAME','SUBSECTION','NTEE'],index_col='EIN')
df2=pd.read_table('crosswalk.txt',sep='|')
df2.columns=['XML','EIN']
df3=pd.read_table('miss.txt',header=None,names=['XML','MISSION','CONTRIBUTION','SERVICE','BEGIN_YEAR','END_YEAR'],na_values=.0000)
df3['YEAR']=df3['BEGIN_YEAR'].str[:4].astype(int)
print('files imported')
df32=pd.merge(df3,df2,how='left',on='XML')
print('merging financial info')
group=df32[['EIN','CONTRIBUTION']].dropna(axis=0,how='any').groupby('EIN')
fin=group.mean()
fin.columns=['CONTRIBUTION']
fin2=group.count()
fin2.columns=['YEARS']
fin3=fin.join(fin2)
df4=df1.join(fin3,how='left')
mis=df32[['EIN','MISSION','YEAR']].sort_values(by='YEAR').drop_duplicates(subset='EIN',keep='last').set_index('EIN')
print('merging mission info')
df5=df4.join(mis,how='left')
print('merging finished,dropping missing',len(df5))
df6=df5.dropna(axis=0,how='any')
df7=df6.drop('YEAR',1)
print('saving to ngo2.txt...total rows',len(df7),df7.columns)
df7.to_csv('ngo2.txt',sep='\t')
```

2. $ Python3 gtopic.py: processing mission statements data for LDA.

1) Tokenize;

2) Remove organizations that have less then 20 words mission statements (207,000+ to 114,000+). This is aimed to make different mission statements comparable in length and structure, and to reduce the computational expense of LDA. Ideally, we could carry out all the analysis below to this left-out subset

and draw some comparisons;

3) Named Entity Recognition and extract only nouns (to reduce the noise brought by verbs and adjectives that are not directly related to social ideas;

4) Lemmatize nouns;

5) Remove trivial words, including nouns that appears only once (usually nonprofits' names) and nouns that appears very often but not informative:

["mission","object","goal","purpose","not-for-profit","nonprofit","ngo","organization","community", "program","activity","service","education","training","volunteer","support","people","need"];

6) Generate dictionaries (word to id) and document-words matrix.

```
print("Loading dataset...")
t0 = time()
data = pd.read_table('ngo2.txt',index_col='EIN')
tokenizer = RegexpTokenizer(r'\w+')
data['COUNT']=data['MISSION'].apply(lambda x: len(tokenizer.tokenize(str(x))))
documents = data[data['COUNT']>20]['MISSION'].tolist()
print("Number of Organizations",len(documents),"done in %0.3fs." % (time() - t0))
stoplist= ["mission","object","goal","purpose","not-for-profit","nonprofit","ngo","organization","community",
"program","activity","service","education","training","volunteer","support","people","need"]
print("Generating distribution")
wnl = WordNetLemmatizer()
fdist=nltk.FreqDist(wnl.lemmatize(word) for doc in documents for word in tokenizer.tokenize(doc))
tf1=fdist.hapaxes()
def nouns(sent):
    nouns=[token for token,pos in pos_tag(word_tokenize(sent)) if pos.startswith('N')]
    return nouns
print("Tokenizing, extrating nouns, removing non-informative words and lemmatizing")
t0 = time()
def tokenize(doc):
    tokens = [word for sent in nltk.sent_tokenize(doc) for word in nouns(sent)] # first tokenize by sentence, then by word to extract nou
    wnl = WordNetLemmatizer()
    singulars=[wnl.lemmatize(t) for t in tokens]
    filtered_tokens = []
    # filter out any tokens not containing letters (e.g., numeric tokens, raw punctuation)
    for token in singulars:
        if (re.search('[a-z]', token) and token not in stoplist and token not in tf1):
            filtered_tokens.append(token)
    return filtered_tokens
documents = [ tokenize(doc.lower()) for doc in documents ]
print("Pre-processing done in %0.3fs." % (time() - t0))
print("Generating dictionary and corpus")
# Sort words in documents
for doc in documents:
    doc.sort()
dictionary = corpora.Dictionary(documents) # Build a dictionary where for each document each word has its own id
dictionary.compactify()
dictionary.save('mission2.dict') # and save the dictionary for future use
corpus = [dictionary.doc2bow(doc) for doc in documents] # Build the corpus: vectors with occurence of each word for each document
corpora.MmCorpus.serialize('mission2.mm', corpus) # and save in Market Matrix format
```

3. $ Python3 lda.py: run LDA topic models to generate document-topic matrix

1) Loop through different number of topics: 25, 50, 100, 200, 300, 500, 1000.

```
from gensim import corpora, models, similarities
print("Loading data")
# Initialize Parameters
corpus_filename = 'mission2.mm'
dict_filename   = 'mission2.dict'

for n in [25,50,100,200,300,500,1000]:
    lda_filename    = 'mission'+str(n)+'.lda'
    lda_params      = {'passes': 5, 'alpha': 'auto'}
    # Load the corpus and Dictionary
    corpus = corpora.MmCorpus(corpus_filename)
    dictionary = corpora.Dictionary.load(dict_filename)

    print("Running LDA with: %s  topics" % n)
    lda = models.LdaModel(corpus, id2word=dictionary,num_topics=n,passes=lda_params['passes'],alpha = lda_params['alpha'])
    lda.save(lda_filename)
    print("lda saved in %s " % lda_filename)
```

4. $ Jupyter Noterbook, pyLDAvis.ipynb: visualizations of topic modeling

1) Run pyLDAvis to visualize the LDA results (25, 50, 100, 200 topics).

In [ ]:
```
from gensim import corpora, models
import pyLDAvis.gensim
```

In [ ]:
```
corpus = corpora.MmCorpus('mission2.mm')
dictionary = corpora.Dictionary.load ('mission2.dict')
```

1.Number of Topics=25

In [ ]:
```
lda25 = models.LdaModel.load('mission25.lda')
mission_data25 =  pyLDAvis.gensim.prepare(lda25, corpus, dictionary)
pyLDAvis.display(mission_data25)
```

2.Number of Topics=50

In [ ]:
```
lda50 = models.LdaModel.load('mission50.lda')
mission_data50 =  pyLDAvis.gensim.prepare(lda50, corpus, dictionary)
pyLDAvis.display(mission_data50)
```

3.Number of Topics=100

In [ ]:
```
lda100 = models.LdaModel.load('mission100.lda')
mission_data100 =  pyLDAvis.gensim.prepare(lda100, corpus, dictionary)
pyLDAvis.display(mission_data100)
```

4.Number of Topics=200

5. $ Python3 feature.py <input> <output> <# topics>: final feature generation LDA+ NTEE.

1) Convert NTEE code to NTEE dummies;

2) Convert contributions to dummies to avoid complications in multi-class classification. In specific, contributions=1 if >$288903.5 million/year (50 percentile in our dataset). This label measures above median vs below median contributions.

3) Convert LDA models to pandas dataframe and merge features.

```
def generate_features(data,topics):
    features = pd.DataFrame(index = data.index)
    features = generate_label(data, features, 1000000)
    features = generate_dummies(data, features)
    features = generate_vectors(features,features,topics)
    return features
def generate_vectors(data,features,n):
    corpus = corpora.MmCorpus('mission2.mm')
    dictionary = corpora.Dictionary.load ('mission2.dict')
    lda = models.LdaModel.load('mission'+str(n)+'.lda')
    print('mission'+str(n)+'.lda is loaded')
    doc = [dict(lda[x]) for x in corpus]
    calc=pd.DataFrame(doc,index = data.index)
    features=features.join(calc)
    return features
def generate_dummies(data,features):
    var=['NTEE']
    for v in var:
        calc = pd.DataFrame(index=data.index)
        new = data[data[v].notnull()][v]
        calc = calc.join(new, how = 'left')
        if v=='NTEE':
            calc[v] = v[:4]+'_'+data[data[v].notnull()][v].astype(str).str.strip().str.get(0)
        else:
            calc[v] = v[:4]+'_'+data[data[v].notnull()][v].astype(str).str.strip()
        calc.fillna(value = v+'_MISSING', inplace = True)
        rv = pd.get_dummies(calc[v])
        features=features.join(rv)
    print('dummies generated')
    return features
def generate_label(data,features,cap):
    calc = pd.DataFrame(index=data.index)
    calc['CONTRIBUTION']=data['CONTRIBUTION'].apply(lambda x: 1 if x > cap else 0)
    print ('contributions label generated')
    return features.join(calc)
def run(inp, out,topics):
    print("Loading dataset...")
    t0 = time()
    data = pd.read_table(inp,index_col='EIN')
    tokenizer = RegexpTokenizer(r'\w+')
    data['COUNT']=data['MISSION'].apply(lambda x: len(tokenizer.tokenize(str(x))))
    data = data[data['COUNT']>20]
    print("Number of Organizations",len(data),"done in %0.3fs." % (time() - t0))
    features = generate_features(data,topics)
    features.to_csv(out,sep='\t')
    print ("Wrote file to {}".format(out))
if __name__ == "__main__":
    print ("This program produces all features for model generation.")
    print ("It will write the file features data to a separate file.")
    print ("Usage: python feature.py <Input> <Output> <number of topics>")
```

6. $ Python3 model.py <input><output>: loop through different classifiers and evaluate them

1) loop through different feature space (25, 50, 100, 200, 300, 500, 1000 topics);

2) for each feature space, loop through 1000+ models/parameters (10 types of models);

3) evaluate the model based on the 90-10 test set;

4) record evaluation data into a table for comparisons between classifiers.

```
def generate_features(data,topics):
    features = pd.DataFrame(index = data.index)
    features = generate_label(data, features, 288903.5)
    features = generate_dummies(data, features)
    features = generate_vectors(features,features,topics)
    return features
def generate_vectors(data,features,n):
    corpus = corpora.MmCorpus('mission2.mm')
    dictionary = corpora.Dictionary.load ('mission2.dict')
    lda = models.LdaModel.load('mission'+str(n)+'.lda')
    print('mission'+str(n)+'.lda is loaded')
    doc = [dict(lda[x]) for x in corpus]
    calc=pd.DataFrame(doc,index = data.index)
    features=features.join(calc)
    return features
def generate_dummies(data,features):
    var=['NTEE']
    for v in var:
        calc = pd.DataFrame(index=data.index)
        new = data[data[v].notnull()][v]
        calc = calc.join(new, how = 'left')
        if v=='NTEE':
            calc[v] = v[:4]+'_'+data[data[v].notnull()][v].astype(str).str.strip().str.get(0)
        else:
            calc[v] = v[:4]+'_'+data[data[v].notnull()][v].astype(str).str.strip()
        calc.fillna(value = v+'_MISSING', inplace = True)
        rv = pd.get_dummies(calc[v])
        features=features.join(rv)
    print('dummies generated')
    return features
def generate_label(data,features,cap):
    calc = pd.DataFrame(index=data.index)
    calc['CONTRIBUTION']=data['CONTRIBUTION'].apply(lambda x: 1 if x > cap else 0)
    print ('contributions label generated')
    return features.join(calc)
def run(inp, out,topics):
    print("Loading dataset...")
    t0 = time()
    data = pd.read_table(inp,index_col='EIN')
    tokenizer = RegexpTokenizer(r'\w+')
    data['COUNT']=data['MISSION'].apply(lambda x: len(tokenizer.tokenize(str(x))))
    data = data[data['COUNT']>20]
    print("Number of Organizations",len(data),"done in %0.3fs." % (time() - t0))
    features = generate_features(data,topics)
    features.to_csv(out,sep='\t')
    print ("Wrote file to {}".format(out))
if __name__ == "__main__":
    print ("This program produces all features for model generation.")
    print ("It will write the file features data to a separate file.")
    print ("Usage: python feature.py <Input> <Output> <number of topics>")
```

7. $ Python3 importance.py: retrieve parameters for selected models

1) Retrieve the feature importance from Random Forest classifier

2) Retrieve the linear parameters from Logistics Regression classifier

```
def rfparameters(df,label,clf):
    features=np.array(df.ix[:, df.columns != label].describe().keys())
    print('Running RF')
    clf.fit(df[features], df[label])
    print('Plotting and Recording')
    importances = clf.feature_importances_
    sorted_idx = np.argsort(importances)[:10]
    padding = np.arange(10) + 0.5
    pl.barh(padding, importances[sorted_idx], align='center')
    pl.yticks(padding, features[sorted_idx])
    pl.xlabel("Relative Importance")
    pl.title("Variable Importance")
    best_features = features[sorted_idx][::-1]
    ddf=pd.DataFrame(data={'Top Features by RF': best_features})
    return pl.savefig('importanceRF.png'), ddf.to_csv('importanceRF.txt',sep='\t')
def lrparameters(df, label, clf):
    features=[i for i in df.columns if not i==label]
    print('Running LR')
    clf.fit(df[features], df[label])
    print('Recording')
    ddf=pd.DataFrame(clf.coef_,columns=df[features].columns.transpose())
    return ddf.to_csv('importanceLR.txt',sep='\t')
clf=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=100, max_features='sqrt',
            max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=5,min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=-1,
            oob_score=False, random_state=None, verbose=0,warm_start=False)
clf2=LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,verbose=0, warm_start=False)
print('Reading data')
df = pd.read_table('feature1000.txt',index_col='EIN')
df = df.fillna(value=0)
label = 'CONTRIBUTION'
rfparameters(df,label,clf)
lrparameters(df,label,clf2)
```

III. Results and discussions

1. General descriptives

The following graph depicts the raw term frequency distribution (after lemmatization) of the mission statements:

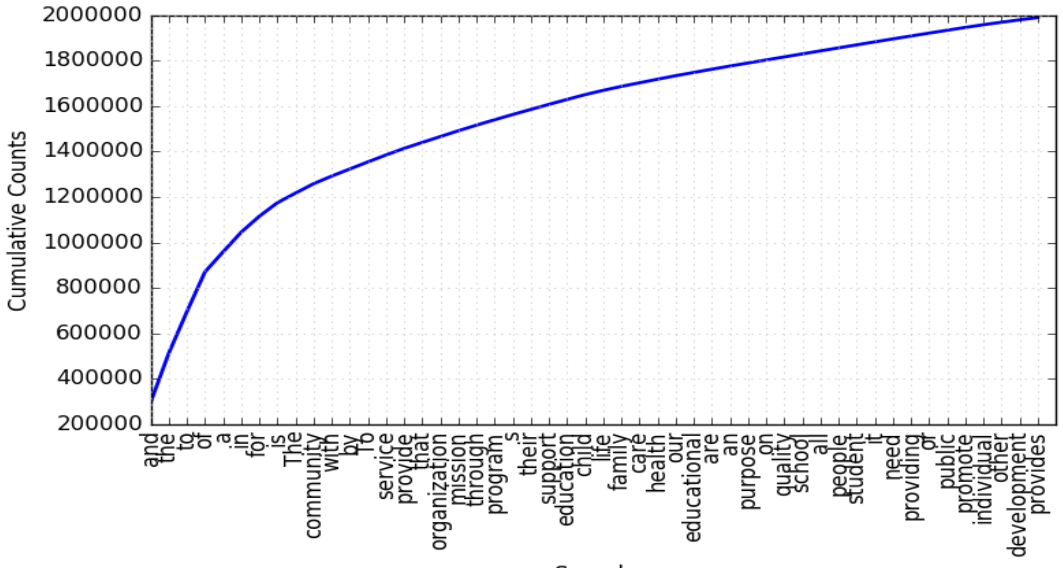


Figure 2 Term frequency distribution of mission statements

Discussions:

- We need to remove stopwords such as “and, the, of,..”;
- We should also remove certain common words because they carry little information about the service types of nonprofits. Words such as service, organization, program, education are almost like cliché in mission statements (see II.2.5);
- Nouns seems to be more informative then other words.

2. Topic modeling[2] (LDA)

2 For an interactive view of some of the topic models, visit my website at mpcs53001.cs.uchicago.edu/~yuxiao/pyLDAvis.html.

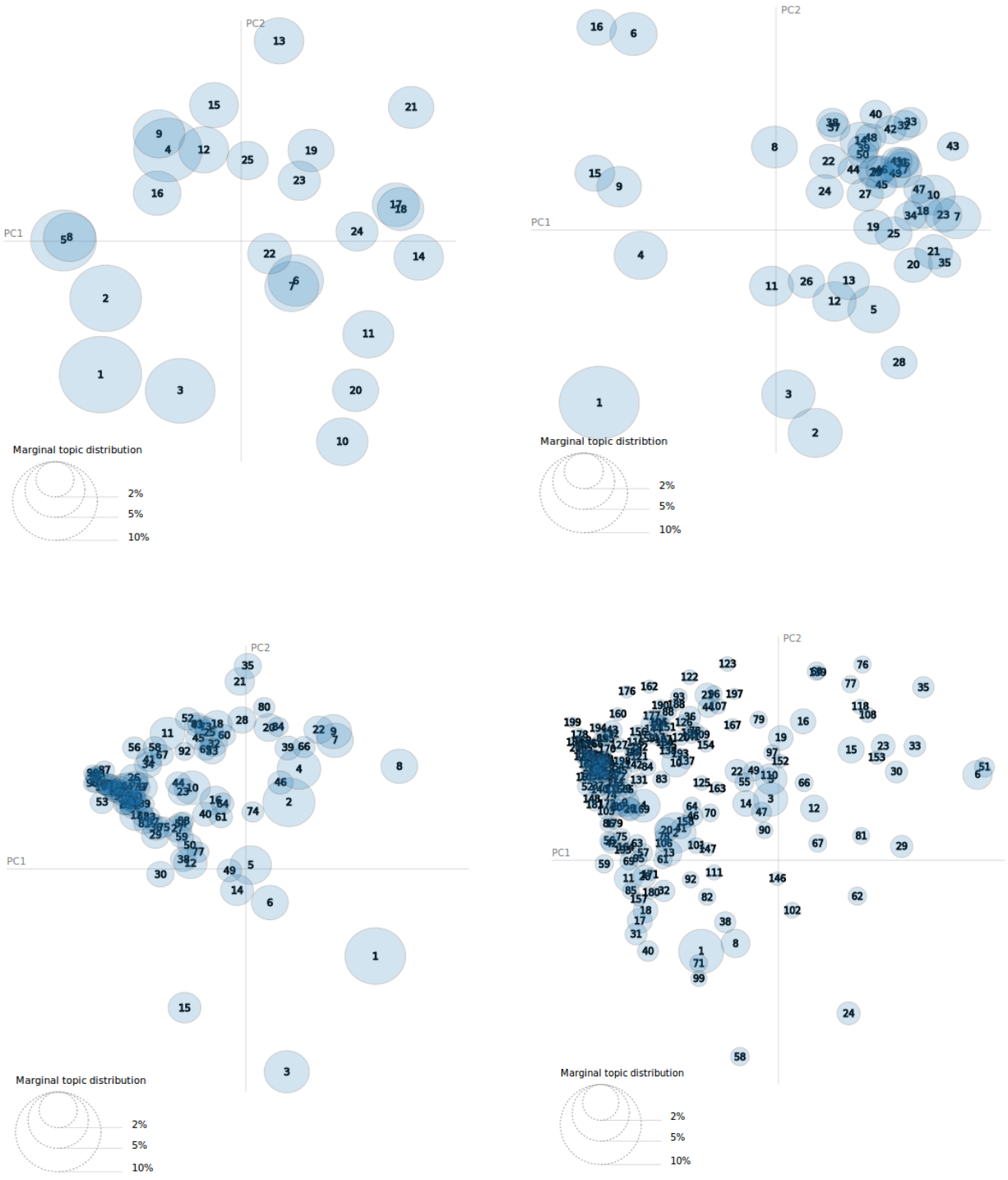


Figure3 Comparisons of different number of topics

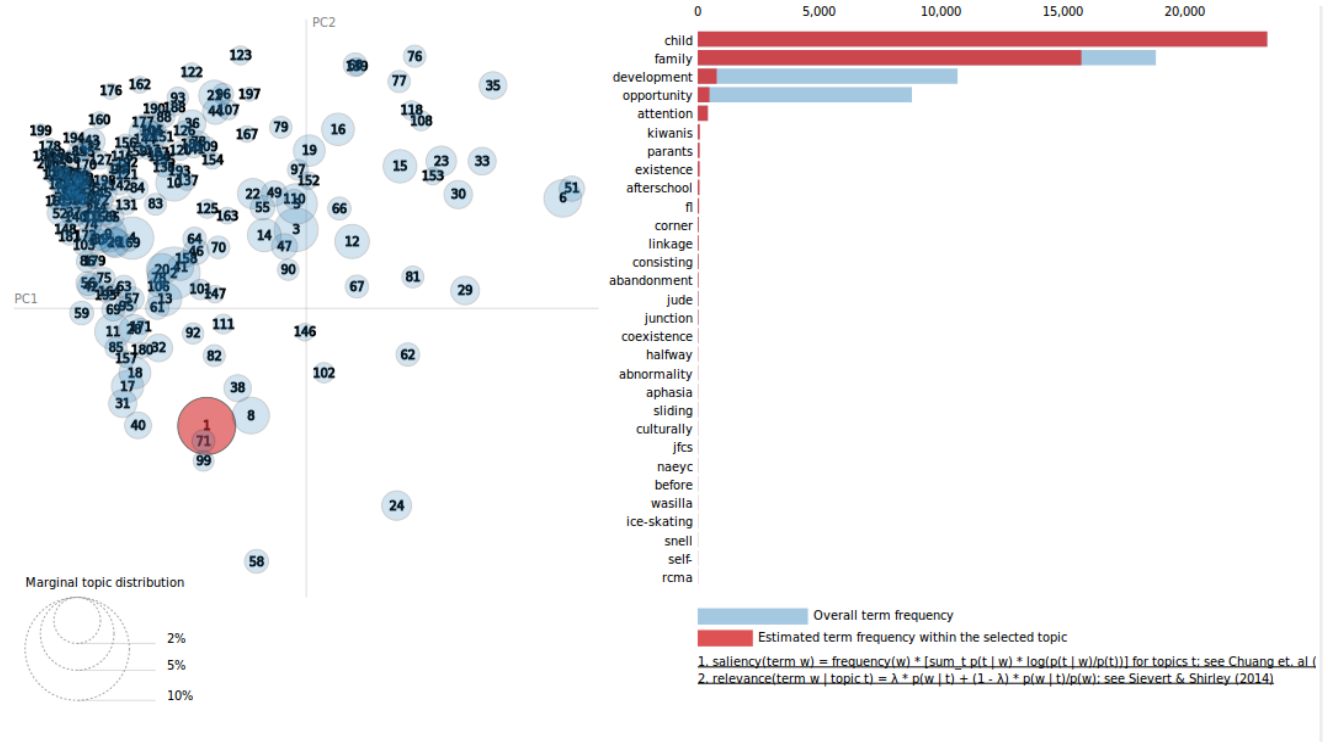


Figure 4 Topic example 1 (child & family), number of topics=200

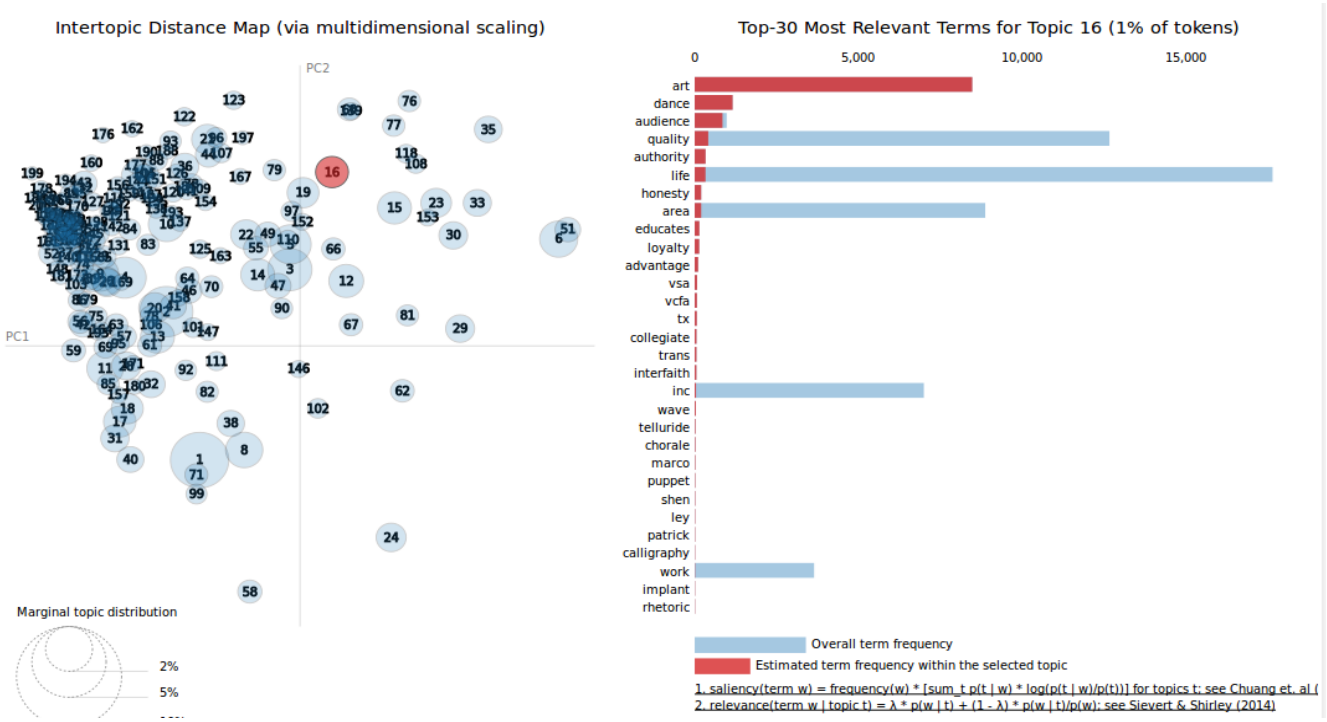


Figure 5 Topic example 16 (art & dance), number of topics=200

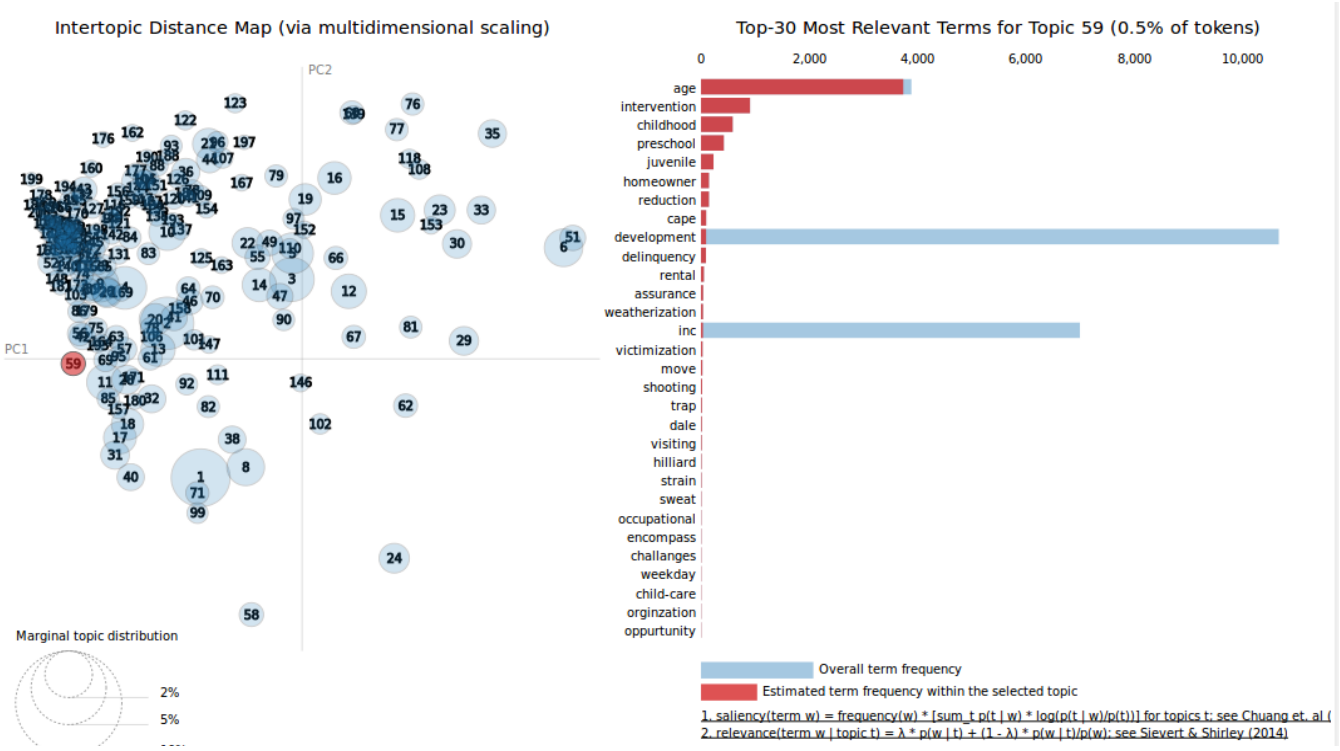


Figure 6 Topic example 59 (youth intervention), number of topics=200

Discussions:

- There seems to be a clusters concentration-drifting from low dimension (25 topics) to high dimension (200 topics);
- The higher the number of topics specified, the small the average cluster size and the smaller set of “characteristics” words per topic;
- Words with highest ratio of estimated term frequency/overall term frequency are usually the 'characteristics” words of a topic;
- To capture a relatively distinct social idea for each topic, that is, to have only 1-2 characteristics words per topic, we need to increase the dimensions. For this project, 200-1000 seems to be a reasonable range for interpretation of social ideas.

3. Classification models

1) Comparisons over feature space

First of all, to find the optimal number of topics used for classification purpose, I run Random Forest classifiers on different feature spaces (number of topics) and get their average performance:

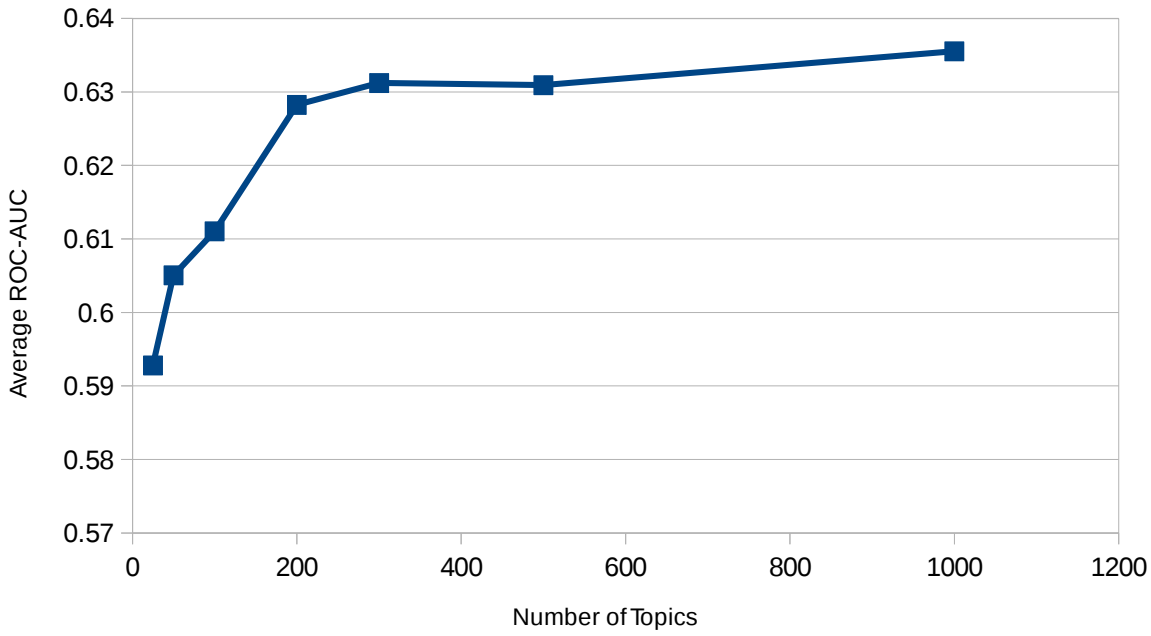


Figure 7 Average ROC-AUC of Random Forest classifiers against number of topics:

Discussions:

- As the number of topics (dimensions) increases, ROC-AUC increases. This is due to the loss of information during LDA transformation, which is just a form of dimension reduction.
- However, the increase of ROC-AUC has diminishing returns, indicating that we could design some information criteria to find the best number of topics (e.g. elbow criteria).

2) Comparisons of classifiers

The above analysis shows that the best feature space is LDA with number of topics=1000. I then apply various models/parameters on this space to predict contributions (median dichotomous):

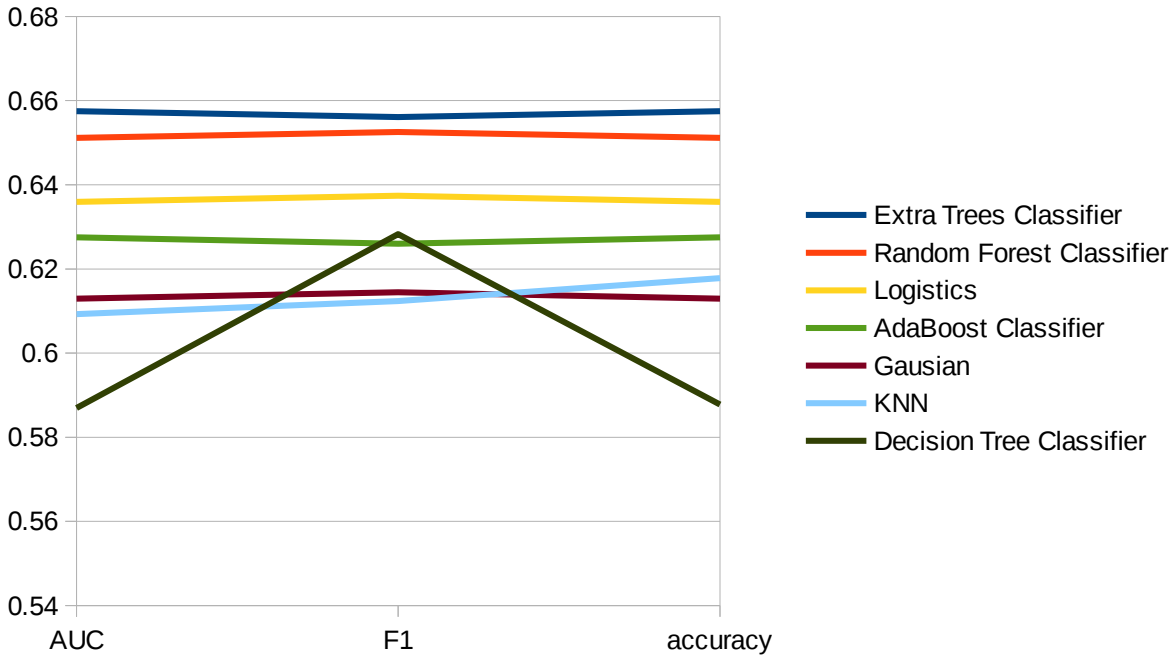


Figure 8 Comparison of different classifiers (number of topics = 1000)

	Random Guess Classifier	Extra Trees Classifier	Increase
AUC	0.5	0.658	31.6%
F1	0.5	0.656	
Accuracy	0.5	0.657	

Table 1 Comparison between the Extra Tree Classifier and Random Guess classifier (50% Threshold)

Discussion:

- In general, boosting methods (random forest, extra trees) works better in almost all the metrics (ROC-AUC, accuracy, F1).

- The best non-boosting classifier is Logistics Regression model.
- Since in our sample, half of the nonprofits have below-median contributions (y=0) and the other half have above-median contributions (y=1). The best baseline model we can use is to guess randomly (50-50). And compare to this base model, with the help of NTEE type and mission statement alone we can increase the model performance by 31.6%. This indicates the predicting power of mission/service type on fund-raising capacity.

3) Interpretations of modeling results

Now we can use the parameters of these models to identify the partial effects of features (social ideas) on fund-raising, which is one of the main purpose of this paper. Below I utilize the Random Forest and Logistics Regression to demonstrate this. The former has a built-in function to extract the importance of "social ideas" based on their information contribution to classify "received contributions". The latter can supply us with the linear parameters associated with each "social idea" indicating the size and sign of their fund-raising capacity.

(1) The hierarchy of social ideas based on optimal Random Forest classifier[3]

3 clf=RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',max_depth=100, max_features='sqrt', max_leaf_nodes=None, min_samples_leaf=1, min_samples_split=5,min_weight_fraction_leaf=0.0, n_estimators=1000, n_jobs=-1,oob_score=False, random_state=None, verbose=0,warm_start=False)

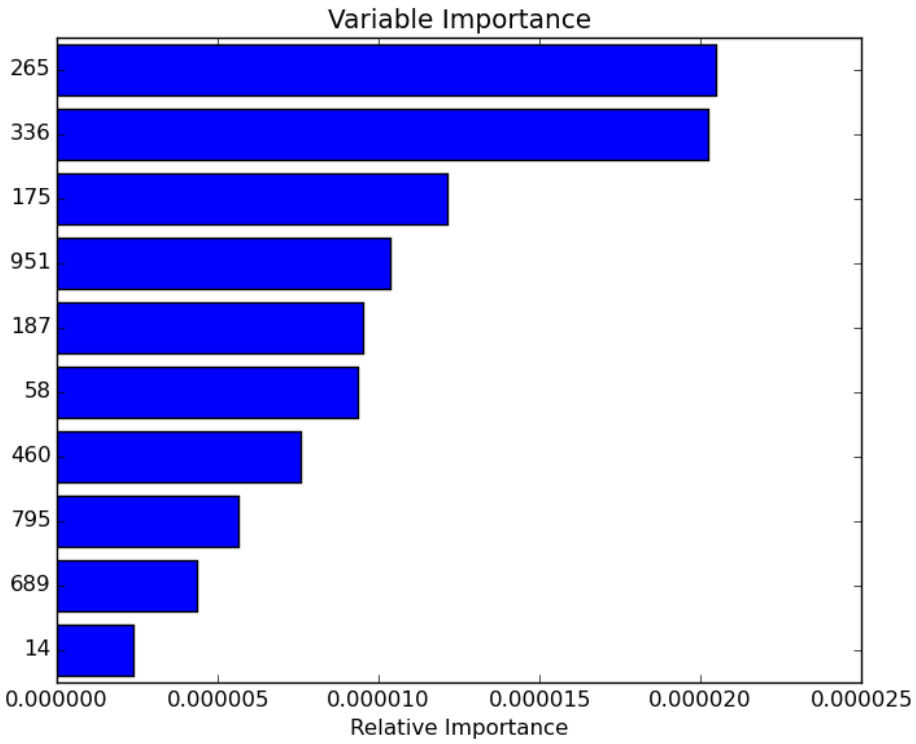


Figure 9 Feature importance from Random Forest classifier

Key words for the top 5 topics:

Topic 265 (family): represent, family, quality

Topic 336 (hiv): hiv, quality, life

Topic 175 (school): school, learning, student

Topic 336 (child development): development, child, family

Topic 951 (places): chatauqua, kansa, opportunity

Discussions:

- These are the top topics that can best discriminate above-median contributions nonprofits and below-median contributions nonprofits. However, as Random Forest is a very complicated

classifier (over 1000 trees in our case), we cannot easily decide the positive/negative impact of these topics/ideas;

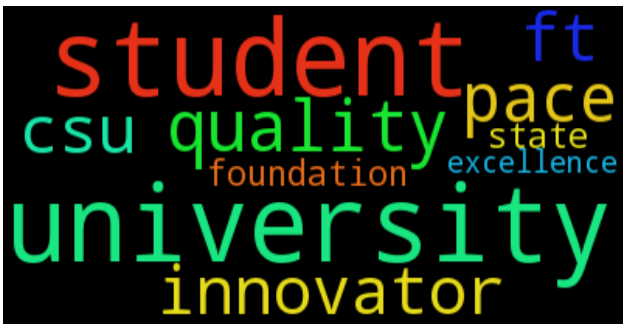
- I noticed that when topics number=1000, most topics has a skewed probability distribution over the words space. The first word usually carries more then 80% of the total weight;
- I have not excluded nouns related to places. Perhaps a better approach is to exclude them in the mission statements and include nonprofits' location information into function (1).

(2) The hierarchy of social ideas based on optimal Logistics Regression classifier[4]

Variable	Parameter	Variable	Parameter
959	4.4114737988	415	-4.483496202
958	4.3009584202	333	-4.043879384
604	4.1818514052	98	-3.626707576
654	4.0950352028	305	-3.582767715
359	4.0688088968	845	-3.576577236
Top 5 positive features		Top 5 negative features	

Words cloud[5] for the top 3 positive topics:

Topic 959: university

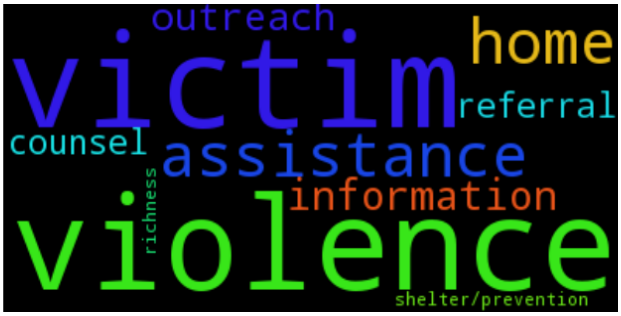


Topic 958: college

4 clf2=LogisticRegression(C=1, class_weight=None, dual=False, fit_intercept=True,intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,penalty='l2', random_state=None, solver='liblinear', tol=0.0001, verbose=0,warm_start=False)

5 Font size indicates probability

Topic 604: crime

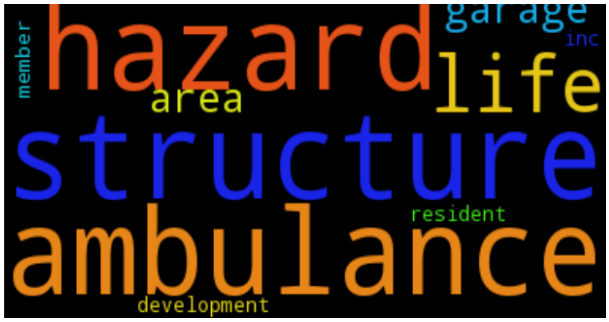


Words cloud for the top 3 negative topics:

Topic 415: school band

Topic 333: a set of values

Topic 98: hazard prevention



Discussion:

- Education (universities and colleges) seems to be the most fund-attracting idea. This is probably due to the tradition of alumni donation. And it is also true in general that educational institutions are among the most “profitable nonprofits” in US;
- One interesting thing to notice is topic 333. It contains a set of values that are usually highly valued, such as “honest”, “loyalty”, “sportsmanship” , “courage”. Yet nonprofits promoting these ideas are poorly supported;
- Notice that both topic 98 (hazard) and topic 604 (crime) seems to be related to public safety. Yet topic 98 is much less popular. This is understandable, as people in general care about short-run problems. And it should be the government's responsibility to take care of our society's long term safety and development;

- For future improvement, I think we could include both nouns and adjectives from the mission statements, but leave out nouns that are not directly related to social ideas (such as location, institution names, etc.).

IV. Conclusions

From a methodology point of view, this paper explores the approach of using topic modeling (LDA in particular) as a way of feature generation for classification models. Ontologically, this paper try to understand collective beliefs of human beings from their collective actions (contributions). The modeling results show the feasibility of this approach and shed some light on the value system of US society. Practically, with more control variables and a better processing of the mission statement (e.g. including adjectives), the government and private foundations could use this model to understand the resource allocation of the nonprofits market and provide subsidies to certain unpopular yet important social ideas and services.