Gendered Pronoun Resolution

Our kenel web site: https://www.kaggle.com/wang540792740/imporving-with-kford

1st Bowen Li
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey 07030
Email: bli50@stevens.edu

2nd Yuanjie Shi
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey 07030
Email: yshi31@stevens.edu

3rd Jiawei Wang
Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, New Jersey 07030
Email: jwang159 @stevens.edu

***Abstract*—Pronoun resolution is part of coreference resolution, the task of pairing an expression to its referring entity. This is an important task for natural language understanding, and the resolution of ambiguous pronouns is a longstanding challenge. Unfortunately, recent studies have suggested gender bias among state-of-the-art coreference resolvers. Therefore, the aim of this project is to build pronoun resolution systems by NLP, Random Forest and K-fold validation. Finally, the whole classifying process will be more efficient**

Keywords—pronoun resolution, NLP, random forest, k-fold validation

I. INTRODUCTION

Natural language processing (NLP) is an important direction in the field of computer science and artificial intelligence. It studies various theories and methods that enable effective communication between humans and computers in natural language. One of the most difficult problems in NLP is semantic disambiguation. In real life, people can understand the semantics and identify the pronouns by referring to the context. But the computer must be trained to complete the task, otherwise it will cause ambiguity.

In this competition, we must identify the target of a pronoun within a text passage. The source text is taken from Wikipedia articles. We are provided with the pronoun and two candidate names to which the pronoun could refer. We must create an algorithm capable of deciding whether the pronoun refers to name A, name B, or neither.

A. Work Breakdown

TABLE 1: Work Breakdown Table

Phase	Activity	Duration	Dependencies	Deliverables
Preparing	T1. Read coursework information and review knowledge T2. Exploratory Data	5 4		M1. Basic Knowledge
Scheduling	T3. Schedule project	2	T1(M1)	M2. Schedule
Requirement	T4. Requirement Plan	2	T2(M2)	M3. Get the Requirement Plan
Analysis	T5. First draft of 2 requirement	2	T3(M3)	M4. Original
	T6. Finish Requirement report	2	T4,5(M4)	M5. Requirement report
	T7. Date Analysis	5	T5, (M5)	M6. Data Analysis
	T8. Feature Analysis	3	T7(M6)	M7. Feature Analysis
	T9. Model Selection and Training	2	T14(M11)	M8. Model Decide
Test	T10. Code Test	2	T7,8,9(M16,7,8)	M9. First test
	T11. Kaggle Perform Test	2	T10(M9)	
	T12. Evaluate Model	1	T10(M9)	M10. Value of loss
Deployment	T13. Check and complete project and report	3	T4,5,6,8,12(M3,4,5,7,10)	Finish. Final report

B. Project Schedule

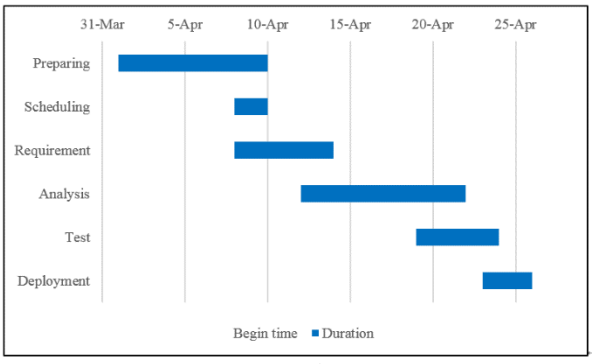


Fig 1: Gantt chart for the Project

C. Risk Management

TABLE 2: Risk Management

Risk Type	Potential Risk	Probability	Effects
Technology	Have problems in Software environment configuration.	High	Tolerable
	Reusable components can't be reused because of their defect.	Moderate	Catastrophic
People	Limited skills of developers	High	Tolerable
	Take the day off	High	Tolerable
Organizational	Communication barriers exist among each person.	Moderate	Serious
	Meeting can't be carried out as scheduled	Moderate	Serious
Requirements	Requirements are changeful and uneasy to notify all team members.	High	Serious
Estimation	Time is limited but project schedules may be delayed for uncertain reasons.	Moderate	Catastrophic

D. Quality Management

Quality management will ensure the level of quality is achieved in a software product. It involves establishing processes and standards. After trade off quality attributes, we think the core feature of our project is efficiency, understandability reusability and security.

II. RELATED WORK

A. Requirement Analyzing

In this competition, we are supposed to achieve a pronoun resolution task based on solving gender bias problem. Specifically, our main task is creating an algorithm to decide which name is the pronoun in the text from Wiki refers to. To solve these problems, we must face these difficulties:

1) Technologies:

In this competition, we need to use kinds of unfamiliar technologies. We are supposed to answer these questions: What technologies do we need in this competition? Which algorithm are we going to use? What is the principle of these technologies? How do we use these technologies in this competition? What does each piece of data in the data set specifically refer to? How to extract features? How do we improve the model?

2) Kaggle:

It is the first time for us to achieve competitions in Kaggle. In order to get a good position on Kaggle, we must understand: What is the basis for Kaggle scoring? What kind of model is most popular on Kaggle? How to use Kaggle's kenel function? How to use Kaggle to improve and improve the model?

B. NLP

Natural language processing (NLP) is a branch of artificial intelligence that helps computers understand, interpret and manipulate human language. NLP draws from many disciplines, including computer science and computational linguistics, in its pursuit to fill the gap between human communication and computer understanding.

1) How does our NLP work?

Natural language processing includes many different techniques for interpreting human language, ranging from statistical and machine learning methods to rules-based and algorithmic approaches. We need a broad array of approaches because the text- and voice-based data varies widely, as do the practical applications. In general terms, NLP tasks break down language into shorter, elemental pieces, try to understand relationships between the pieces and explore how the pieces work together to create meaning.

2) These underlying tasks are often used in higher-level NLP capabilities:

- Content categorization. A linguistic-based document summary, including search and indexing, content alerts and duplication detection.
- Topic discovery and modeling. Accurately capture the meaning and themes in text collections and apply advanced analytics to text, like optimization and forecasting.
- Contextual extraction. Automatically pull structured information from text-based sources.
- Sentiment analysis. Identifying the mood or subjective opinions within large amounts of text, including average sentiment and opinion mining.
- Speech-to-text and text-to-speech conversion. Transforming voice commands into written text, and vice versa.
- Document summarization. Automatically generating synopses of large bodies of text.
- Machine translation. Automatic translation of text or speech from one language to another.

3) NLP methods and applications

Natural language processing goes hand in hand with text analytics, which counts, groups and categorizes words to extract structure and meaning from large volumes of content. Text analytics is used to explore textual content and derive new variables from raw text that may be visualized, filtered, or used as inputs to predictive models or other statistical methods.

4) NLP and text analytics are used together for many applications:

- Investigative discovery. Identify patterns and clues in emails or written reports to help detect and solve crimes.
- Subject-matter expertise. Classify content into meaningful topics so you can take action and discover trends.
- Social media analytics. Track awareness and sentiment about specific topics and identify key influencers.
- Avoid combining SI and CGS units, such as current in

5) The evolution of NLP:

The evolution of NLP toward NLU has a lot of important implications for businesses and consumers alike. Imagine the

power of an algorithm that can understand the meaning and nuance of human language in many contexts, from medicine to law to the classroom. As the volumes of unstructured information continue to grow exponentially, we will benefit from computers' tireless ability to help us make sense of it all.

C. Random forest

In machine learning, a random forest is a classifier that contains multiple decision trees, and the category of its output is determined by the mode of the category of the individual tree output. Specifically, a forest is built in a random way. There are many decision trees in the forest. There is no correlation between each decision tree in the random forest. After getting the forest, when a new input sample enters, let each decision tree in the forest make a separate judgment to see which class the sample should belong to (for the classification algorithm), and then see which One type is selected the most, and the sample is predicted to be that type.

The advantages of random forests are:

- For a wide variety of materials, it can produce high accuracy classifiers;
- It can handle a large number of input variables;
- It can assess the importance of variables when determining categories;
- When constructing a forest, it can internally produce an unbiased estimate of the error after generalization;
- It contains a good way to estimate missing data and maintain accuracy if a large portion of the data is lost;
- It provides an experimental method to detect variable interactions;
- For an unbalanced classification data set, it can balance the error;

1) *Construct tree algorithm:*

- N is used to indicate the number of training cases (samples), and M is the number of features.
- Enter the number of features m to determine the decision result of a node on the decision tree; where m should be much smaller than M.
- From the N training cases (samples), the samples are sampled N times to form a training set (ie, bootstrap sampling), and the unused use cases (samples) are used for prediction to evaluate the error.
- For each node, m features are randomly selected, and the decision of each node on the decision tree is determined based on these features. According to these m features, calculate the best split mode.

The random forest classification effect (error rate) is related to two factors:

- Correlation of any two trees in the forest: the greater the correlation, the greater the error rate;
- The ability to classify each tree in the forest: The stronger the classification ability of each tree, the lower the error rate of the entire forest.

By reducing the number of feature selections m, the correlation and classification ability of the tree will be reduced accordingly; increasing m will increase the two. So the key question is how to choose the optimal m (or range), which is the only parameter of the random forest. To solve this problem, it is mainly based on the out-of-bag error. It is calculated as follows:

- For each sample, calculate its classification as a tree of oob samples (approximately 1/3 of the tree);
- Then vote with a simple majority as the classification result of the sample;
- Finally, the ratio of the number of misclassifications to the total number of samples is used as the oob misclassification rate of the random forest.

2) *scikit-learn using method:*

In scikit-learn, Random Forest's classification class is Random Forest Classifier, and the regression class is Random Forest Regressor. The parameters to be parameterized include two parts. The first part is the parameters of the Bagging framework, and the second part is the parameters of the CART decision tree. The function is:

$$\begin{aligned}\text{RandomForestClassifier}(&\text{n\_estimators} = 10, \text{criterion} \\ &= \text{'gini'}, \text{max\_depth} \\ &= \text{None}, \text{min\_samples\_split} \\ &= 2, \text{min\_samples\_leaf} \\ &= 1, \text{min\_weight\_fraction\_leaf} \\ &= 0.0, \text{max\_features} \\ &= \text{'auto'}, \text{max\_leaf\_nodes} \\ &= \text{None}, \text{min\_impurity\_decrease} \\ &= 0.0, \text{min\_impurity\_split} \\ &= \text{None}, \text{bootstrap} = \text{True}, \text{oob\_score} \\ &= \text{False}, \text{n\_jobs} = 1, \text{random\_state} \\ &= \text{None}, \text{verbose} = 0, \text{warm\_start} \\ &= \text{False}, \text{class\_weight} = \text{None}\end{aligned}$$

The parameter description of the constructor is:

- N_estimators: numeric value.
  The number of decision trees in the forest, the default is 10
- Criterion: character value
  Which method is used to measure the quality of splitting, information entropy or Gini index, the default is the Gini index
- Max_features: values are int, float, string, or None(), default "auto"
  The number of features to be considered when seeking the best segmentation, that is, when the number of features is large, is divided.
  Int:max_features is equal to this int value
- Float:max_features is a percentage, and each (max_features * n_features) feature is considered in each split.

"auto": max_features is equal to sqrt(n_features)
"sqrt": equal to "auto"
"log2": max_features=log2(n_features)
None:max_features = n_features

- Max_depth: Int type value or None, default is None
  Maximum depth of the tree
- Min_samples_split: Int type value, float type value, default is 2
  Minimum number of samples required to split internal nodes
  Int: if it is an int value, it is this int value
  Float: min_samples_split * n_samples if it is a float value
- Min_samples_leaf: int value, float value, default is 1
  Sample minimum contained on the leaf node
  Int: is this int value
  Float:min_samples_leaf * n_samples
- Min_weight_fraction_leaf : float,default=0.
  The condition that can become a leaf node is that the ratio of the number of instances corresponding to the node to the total number of samples is at least greater than the value of the min_weight_fraction_leaf.
- Max_leaf_nodes: int type, or None (default None)
  The maximum number of leaf nodes, the tree is generated in the best priority. The best node is defined as a relatively low impurity, that is, a leaf node with higher purity.
- Min_impurity_split:float value
  The threshold for the growth of the tree to stop. A node will split if its impurity level is greater than this threshold; if it is lower than this value, it will become a leaf node.
- Min_impurity_decrease: The value of float is 0. The default is 0.
  A node will be split, and if it is split, the effect of reducing the impurity is higher than this value.
- Bootstrap: boolean type value, default True
  Whether to use a reversible sampling method

D. K-fold Cross-Validation

K-fold cross-validation, the initial sampling is divided into K sub-samples, a single sub-sample is retained as the data of the verification model, and the other K-1 samples are used for training. Cross-validation is repeated K times, each sub-sample is verified once, and the average K-time results or other combinations are used to finally obtain a single estimate. The advantage of this method is that it repeatedly uses randomly generated subsamples for training and verification. Each time the results are verified once, 10-fold cross-validation is the most commonly used.

The steps are:

- Divide all training sets S into k disjoint subsets. Assuming that the number of training examples in S is m, then each subset has m/k training examples, and the corresponding subset is called $\{S_1, S_2, \dots, S_k\}$.
- Take M_i out of the model set M each time, and then select k-1 datasets $\{S_1, S_2, \dots, S_{j-1}, S_{j+1}, \dots, S_k\}$ in the training subset. (That is, only S_j is left at a time), after training M_i with these k-1 subsets, the hypothesis function h_{ij} is obtained. Finally, use the remaining S_j for testing and get an empirical error $\hat{\varepsilon}_{S_j}(h_{ij})$.
- Since we leave S_j (j is from 1 to k) each time, we get k empirical errors, then for M_i, its empirical error is the average of these k empirical errors.
- Select the lowest average error rate of M_i, and then use all the S to do another training to get the final h_i.

In scikit-learn, we use follow codes to achieve K-fold cross-validation:

```
import numpy as np
from sklearn.model_selection import KFold
X = ["a", "b", "c", "d"]
kf = KFold(n_splits = 2)
for train, test in kf.split(X):
print("%s %s" % (train, test))
```

III. SOLUTIONS

A. NLP Implement

The template is used to format your paper and style the text. All margins, column widths, line spaces, and text fonts are prescribed; please do not alter them. You may note peculiarities. For example, the head margin in this template measures proportionately more than is customary. This measurement and others are deliberate, using specifications that anticipate your paper as one part of the entire proceedings, and not as an independent document. Please do not revise any of the current designations.

TABLE 3: Header of Dataset

Column	Header	Description
1	ID	Unique identifer for an example (two pairs)
2	Text	Text containing the ambiguous pronoun and two candidate names. About a paragraph in length
3	Pronoun	The pronoun, text
4	Pronoun-offset	Character offset of Pronoun in Column 2 (Text)
5	A	The first name, text
6	A-offset	Character offset of A in Column 2 (Text)
7	A-coref	Whether A corefers with the pronoun, TRUE or FALSE
8	B	The second name, text
9	B-offset	Character offset of B in Column 2 (Text)
10	B-coref	Whether B corefers with the pronoun, TRUE or FALSE
11	URL	The URL of the source Wikipedia page

To exacting features, we use following methods:

- Firstly, we fill ‘NaN’ value with ‘-1’ by using function ’fillna(-1)’.
- Then we get features offset means the distance between the first letter of the first word and the first letter of the first pronoun. Consider that the lens of the pronoun can influence of the result, we add the lens of the pronoun to offset and set it to ‘offset2’. So we decided to use the distance between pronoun and name as our feature.
- Add feature to a new dataframe.
- Finally, we add the NLP features.

Get NLP features:

- Split text to single words.
- Use token to separate words and obtain the dependent relationship.
- Set possessor as the main variable.
- Apply possessor analysis into two different subjects.
- Add new variables “A-poss” and “B-poss” to the data frame.

B. *RandomForestClassifier Implement*

model=multiclass.OneVsRestClassifier(ensemble.RandomForestClassifier(max_depth=7,n_estimators=1000, random_state=33)):

- Max_depth: The maximum depth of the decision tree. If not entered, the decision tree does not limit the depth of the subtree when building the subtree. In the case of a large sample size and many features, the maximum depth needs to be limited. The specific value depends on the distribution of the data.
- N_estimators: The maximum number of iterations of the weak learner, or the maximum number of weak learners. In general, n_estimators are too small, easy to fit, n_estimators is too large, the amount of calculation will be too large, and after n_estimators reaches a certain number, the model increase obtained by increasing n_estimators will be small.
- Random_state: This parameter makes the result easy to reproduce. A certain random value will produce the same result, with the parameters and training data unchanged.

C. *Result Testing*

Rule: Each team can select up to 2 submissions to be used to count towards your final leaderboard score. Submissions that will most likely be best overall, and not necessarily on the public subset.

D. *Kernel Testing*

Data Source: Based on dataset in Kaggle and private dataset in ‘gap-answers’, we enlarge our dataset to get a more general result to test out model.

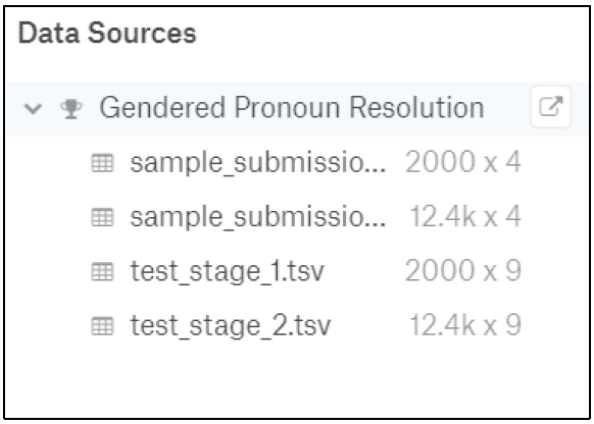


Fig 2: Dataset

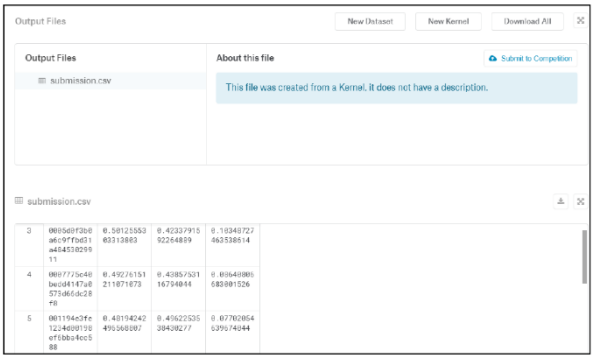


Fig 3: Output

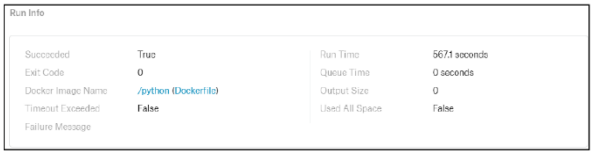


Fig 4: Run Information of Kernel

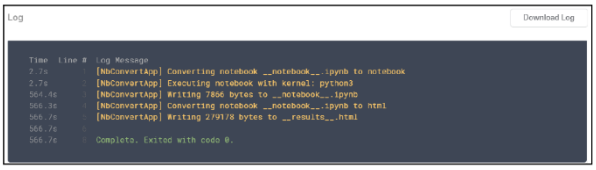


Fig 5: Log of Kernel

E. *Kaggle Leaderboard*

Fig 6: Score

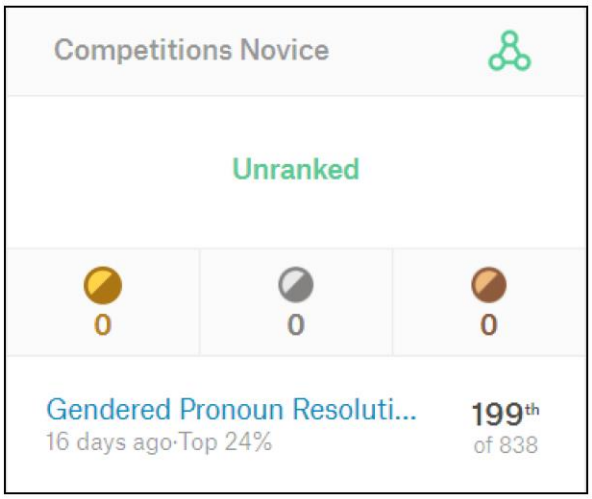


Fig 7: Rank

IV. COMPARISON

We have try many methods from Kaggle Kernel like CNN, and RNN model. Detailed comparisons are shown below:

A. *CNN*

Convolutional neural network (convolutional neural network, CNN or ConvNet) is a feedforward neural network with local connectivity, weight sharing and other characteristics. CNN can capture local features and can function as an n-gram in natural language processing. By combining the captured local features, the combined result can be seen as a textual representation of the input text, in which the text is represented. Based on the various NLP tasks can be completed. The overall framework and its application in terms of images are not much different.

The main disadvantages are:

- CNN needs to use different sizes of windows for the word vector of all words in the sentence to obtain certain context information. But there are some empirical problems selected on the window size. And the problem of text length dependence is not well solved because the window size generally do not choose a big one.
- CNN cannot achieve better analysis and utilization of context semantics.

B. *RNN*

The Recurrent Neural Network Model (RNN) is an artificial neural network in which nodes are connected in a loop. It is a feedback neural network. RNN uses internal memory to process input sequences of arbitrary timing and has internals between its processing units. The feedback connection has a feedforward connection, which makes it easier for RNN to process unfragmented text.

The main disadvantages are:

- Since RNN can only memorize part of the sequence, it is far worse than the short sequence on the long sequence, resulting in a decrease in accuracy once the sequence is too long.
- RNN is easier to overfitting.

V. FUTURE IMPROVEMENT

We have introduced a novel feature for pronoun resolution and demonstrated its significant contribution to pronoun resolution system. This feature aids coreference decisions in many situations not handled by traditional coreference systems. Also, by analysis the features, we are able to build the most complete and accurate table of probabilistic gender information. In the future, we can improve the features and models by using some other algorithms like XGBoost and BERT.

A. *XGBoost*

XGBoost is one of the most popular and efficient implementations of the Gradient Boosted Trees algorithm, a supervised learning method that is based on function approximation by optimizing specific loss functions as well as applying several regularization techniques.

There are two reasons to use XGBoost are also the two goals of our future project:

- Execution Speed
  Generally, XGBoost is fast. Really fast when compared to other implementations of gradient boosting. Szilard Pafka performed some objective benchmarks comparing the performance of XGBoost to other implementations of gradient boosting and bagged decision trees. He wrote up his results in May 2015 in the blog post titled “Benchmarking Random Forest Implementations”.

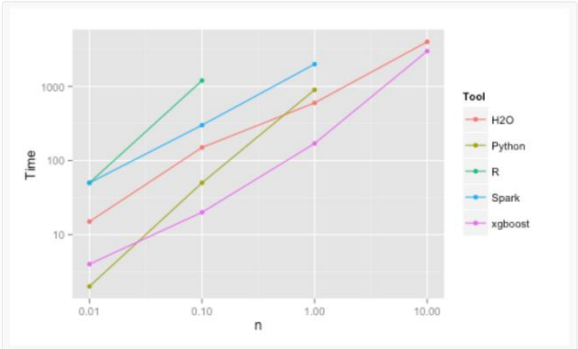


Fig 8: Benchmark Performance of XGBoost

- Model Performance
  XGBoost can be seen to directly take the bias-variance tradeoff into consideration during fitting. This helps the neighborhoods are kept as large as possible in order to avoid increasing variance unnecessarily, and only made smaller when complex structure seems apparent. While coming to high dimensional problem, XGBoost “beats” the curse of dimensionality by not relying on any distance metric and the similarity between data points are learnt from the data through

adaptive adjustment of neighborhoods. This makes the model immune to the curse of dimensionality and deeper trees help to capture the interaction of the features. Thus, there will be no need to search for appropriate transformations.

B. BERT

BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering, Natural Language Inference (MNLI), and others.

BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

BERT is undoubtedly a breakthrough in the use of Machine Learning for Natural Language Processing. The fact that it's approachable and allows fast fine-tuning will likely allow a wide range of practical applications in the future. In this summary, we attempted to describe the main ideas of the paper while not drowning in excessive technical details. For those wishing for a deeper dive, we highly recommend reading the full article and ancillary articles referenced in it. Another useful reference is the BERT source code and models, which cover 103 languages and were generously released as open source by the research team.

Except for the above algorithms, we can also apply some traditional deep learning methods to do feature extractions and model training. For example, CNN, RNN and Gensim library.

VI. CONCLUSION

A. About us

There are 40 days in our team. However, teamwork may cause some problems, such as communication problems. 2 of us are from Computer Engineering, one of us is from Electronic Engineering. Different majors are beneficial to the division of work. Makes individual work clearer and more reasonable. The whole project is based on the agile software development. This process makes our work efficient, because everyday goal is clear. We barely don't waste time in meaningless discussion. This development process is not only highly efficient, but also flexible. For example, once we have a new idea, we can directly discuss, make conclusions and execute. I suppose that this is a valuable experience for every member in the team.

B. About our Project

The project aimed at getting the correct classify of gendered pronoun, which can be missed according to the wrong text or misunderstanding of the sentence. We also meet many problems. For example, the feature is really hard for us to select and what really puzzled us is that what exactly way we use. As for us, we don't use normal ways. We finally choose Random Forest model to achieve our aim.

C. Challenge

- Unfamiliar with NLP (nature language processing), so that need to find more reference to know about this part of knowledge. At the same time, we need to try to use NLP to deal with the complicated task.
- That is our first time to use Random forest function, then we need to figure out the parameter of the function. Especially how to choose the best parameter and compared with other algorithms, such as Decision Tree model.

REFERENCES

[1] D. Albanese, G. Merler, S.and Jurman, and R. Visintainer. MLPy: high-performance python package for predictive modeling. In NIPS, MLOSS Workshop, 2008. J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[2] C.C. Chang and C.J. Lin. LIBSVM: a library for support vector machines. http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.

[3] P.F. Dubois, editor. Python: Batteries Included, volume 9 of Computing in Science & Engineering. IEEE/AIP, May 2007.

[4] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. LIBLINEAR: a library for large linear classification. The Journal of Machine Learning Research, 9:1871–1874, 2008.

[5] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. Journal of Statistical Software, 33(1):1, 2010.

[6] I Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror. Result analysis of the NIPS 2003 feature selection challenge, 2004.

[7] M. Hanke, Y.O. Halchenko, P.B. Sederberg, S.J. Hanson, J.V. Haxby, and S. Pollmann. PyMVPA: A Python toolbox for multivariate pattern analysis of fMRI data. Neuroinformatics, 7(1):37–53, 2009. 2829 PEDREGOSA, VAROQUAUX, GRAMFORT ET AL.

[8] T. Hastie and B. Efron. Least Angle Regression, Lasso and Forward Stagewise. http://cran.r-project.org/web/packages/lars/lars.pdf, 2004.

[9] V. Michel, A. Gramfort, G. Varoquaux, E. Eger, C. Keribin, and B. Thirion. A supervised clustering approach for fMRI-based inference of brain states. Patt Rec, page epub ahead of print, April 2011. doi: 10.1016/j.patcog.2011.04.006.

[10] K.J. Milmann and M. Avaizis, editors. Scientific Python, volume 11 of Computing in Science & Engineering. IEEE/AIP, March 2011.

[11] S.M. Omohundro. Five balltree construction algorithms. ICSI Technical Report TR-89-063, 1989.

[12] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. SIAM Journal on Matrix Analysis and Applications, 31(3):1100–1124, 2009.

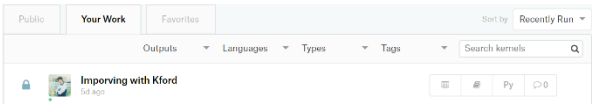
[13] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Ruckstieß, and J. Schmidhuber. ¨ PyBrain. The Journal of Machine Learning Research, 11:743–746, 2010.

[14] S. Sonnenburg, G. Ratsch, S. Henschel, C. Widmer, J. Behr, A. Zien, F. de Bona, A. Binder ¨ , C. Gehl, and V. Franc. The SHOGUN machine learning toolbox. Journal of Machine Learning Research, 11:1799–1802, 2010.

[15] S. Van der Walt, S.C Colbert, and G. Varoquaux. The NumPy array: A structure for efficient numerical computation. Computing in Science and Engineering, 11, 2011.

[16] T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for data processing (MDP): A Python data processing framework. Frontiers in Neuroinformatics, 2, 2008.」

APPENDIX A CODE OF THE SYSTEMS



In [1]
```
import pandas as pd
import spacy
from sklearn.model_selection import KFold
import numpy as np
nlp = spacy.load('en_core_web_sm')
from sklearn import *
import warnings
warnings.filterwarnings('ignore')



test = pd.read_csv('../input/test_stage_2.tsv',delimiter='\t').
rename(columns={'A': 'A_Noun', 'B': 'B_Noun'})
sub=pd.read_csv('../input/sample_submission_stage_2.csv'
)
test.shape, sub.shape
```

Out[1]
```
((12359, 9), (12359, 4))
```

In [2]
```
gh_test=pd.read_csv("https://raw.githubusercontent.com/g
oogle-research-datasets/gap-coreference/master/gap-
test.tsv", delimiter='\t')
gh_valid=pd.read_csv("https://raw.githubusercontent.com/
google-research-datasets/gap-coreference/master/gap-
validation.tsv", delimiter='\t')
train = pd.concat((gh_test,gh_valid)).rename(columns={'A'
: 'A_Noun', 'B': 'B_Noun'}).reset_index(drop=True)
```

In [3]
```
def name_replace(s, r1, r2):
    s = str(s).replace(r1, r2)
    for r3 in r1.split(' '):
        s = str(s).replace(r3, r2)
    return s



def get_features(df):
    df['section_min'] = df[['Pronoun-offset', 'A-offset', 'B-
offset']].min(axis=1)
    df['Pronoun-offset2'] = df['Pronoun-offset'] +
df['Pronoun'].map(len)
    df['A-offset2'] = df['A-offset'] + df['A_Noun'].map(len)
    df['B-offset2'] = df['B-offset'] + df['B_Noun'].map(len)
    df['section_max'] = df[['Pronoun-offset2', 'A-offset2', 'B-
offset2']].max(axis=1)
    df['Text'] = df.apply(lambda r: name_replace(r['Text'],
r['A_Noun'], 'subjectone'), axis=1)
    df['Text'] = df.apply(lambda r: name_replace(r['Text'],
r['B_Noun'], 'subjecttwo'), axis=1)


    df['A-dist'] = (df['Pronoun-offset'] - df['A-offset']).abs()
    df['B-dist'] = (df['Pronoun-offset'] - df['B-offset']).abs()
    return (df)


train = get_features(train)
test = get_features(test)


def get_nlp_features(s, w):
    doc = nlp(str(s))
    # print(doc)
    tokens = pd.DataFrame([[token.text, token.dep_] for
token in doc], columns=['text', 'dep'])
    # print(tokens)
    # token.text is the word, token.dep is the characteristic of
a word
    # print("Noun phrases:", [chunk.text for chunk in
doc.noun_chunks])
    # print("Verbs:", [token.lemma_ for token in doc if
token.pos_ == "VERB"])
    # tokens == 'poss' means possessive.
    return len(tokens[((tokens['text']==w) &
(tokens['dep']=='poss'))])
train['A-poss'] = train['Text'].map(lambda x:
get_nlp_features(x, 'subjectone'))
# print(train['A-poss'])
# 2453    2
# Name: A-poss, Length: 2454, dtype: int64


train['B-poss'] = train['Text'].map(lambda x:
get_nlp_features(x, 'subjecttwo'))
test['A-poss'] = test['Text'].map(lambda x:
get_nlp_features(x, 'subjectone'))
test['B-poss'] = test['Text'].map(lambda x:
get_nlp_features(x, 'subjecttwo'))


train = train.rename(columns={'A-coref':'A', 'B-coref':'B'})
train['A'] = train['A'].astype(int)
train['B'] = train['B'].astype(int)
train['NEITHER'] = 1.0 - (train['A'] + train['B'])


col = ['Pronoun-offset', 'A-offset', 'B-offset', 'section_min',
'Pronoun-offset2', 'A-offset2', 'B-offset2', 'section_max', 'A-
poss', 'B-poss', 'A-dist', 'B-dist']
x1, x2, y1, y2 =
model_selection.train_test_split(train[col].fillna(-1),
train[['A', 'B', 'NEITHER']], test_size=0.2,
random_state=1)
print(x1.head())
```

	Pronoun-offset	A-offset	B-offset	...	B-poss	A-dist	B-dist
1699	448	379	390	...	0	69	58
1790	374	315	346	...	2	59	28
1665	349	297	329	...	0	52	20
898	299	262	281	...	1	37	18
270	431	336	379	...	0	95	52

[5 rows x 12 columns]

In [4]
```
model =
multiclass.OneVsRestClassifier(ensemble.RandomForestCl
assifier(max_depth = 7, n_estimators=1000,
random_state=33))


folds = 5
xchange = 0.94
kf = KFold(n_splits=folds, shuffle=False,
random_state=11)
trn = train[col].fillna(-1)
```

```
val = train[["A", "B", "NEITHER"]]
scores = []

for train_index, test_index in kf.split(train):
    x1, x2 = trn.iloc[train_index], trn.iloc[test_index]
    y1, y2 = val.iloc[train_index], val.iloc[test_index]

    model.fit(x1, y1)
    score = metrics.log_loss(y2, model.predict_proba(x2))
    if score < xchange:
        print("log-loss:", score)
    scores.append(score)

# print("CV Score(log-loss):", np.mean(scores))
model.fit(x1, y1)

# print('log_loss', metrics.log_loss(y2,
model.predict_proba(x2)))
model.fit(train[col].fillna(-1), train[['A', 'B', 'NEITHER']])
results = model.predict_proba(test[col])
test['A'] = results[:,0]
test['B'] = results[:,1]
test['NEITHER'] = results[:,2]
test[['ID', 'A', 'B', 'NEITHER']].to_csv('submission.csv',
index=False)
log-loss: 0.9004897845982024
log-loss: 0.9185686778504919
```