Automatic Knowledge Extraction for News Articles

Mengtian Jin[1], Yilong Li[2], and Linying Yang[1]

[1]Institute for Computational and Mathematical Engineering, Stanford University
[2]Department of Computer Science, Stanford University
{mtjin, yilong, lilyy}@stanford.edu

I. Motivation and Tasks

A structured knowledge database of news articles is very important in a variety of areas like news agencies, media websites and some knowledge-based decision making systems. It can be used to recommend customized news articles for users, analyze development of a business by synthesizing similar articles. In this project, we aim at building an interface that will use NLP to automatically relate articles and knowledge by topic. We aim to have a system that integrate news into a predefined knowledge base, based on a database of tagged news articles, and to identify key entities, topics and even high level relationships between articles automatically is important.

Before integrating news articles into a predefined knowledge base, there are some basic tasks for the news dataset, and these are what we will do in this project:

A. General Topic Recognition

For this dataset and this task, there are basically two types of topics we need to get for each new article: the "topic" of the news, which is the broader category of the news like sports, technologies, and world news; and the "theme" of the news, which is the specific topic an article is about, like hurricane, gun control, Nobel prize, etc. One thing we should note is that there can be multiple tags for a single article, for example, the article *US judge orders temporary pause on deportations of reunited migrant families* can be related to tags like "Immigration" "Law & Justice" and "World News". Also we might need to recognize some specific themes mentioned in the article, which may be useful for news recommendation.

B. Named Entity Extraction and Linking

For news categorization we also need to **extract the key named entities** in the article and divide them into different categories like *Places*, *Regions*, *Persons* and *Groups*. There are usually a lot of named entities included in this article but what we need is only the key entities for article tagging. Also for one word or phrase, it can refer to different named entities based on its contexts. So it will be also important to **link the named entity** we extracted to the semantic unit itself.

II. Related Works

A. Named Entity Extraction

Handcraft rules and machine learning algorithms can classify techniques of entity extraction. Typical rule-based systems like SystemT [1] and GATE [2] make use of rules throughout the entire flow. Due to many special cases, handcraft rules are not sufficient and time-consuming. Supervised learning approaches such as HMM, Decision Trees, SVM and ME are current dominant techniques. However, the crucial dependence on the availability of large, representative and high-quality labeled training datasets makes building large-scale NER systems hard. Thus, researches on applications of semi-supervised learning and unsupervised learning without the prerequisite of an annotated corpus are more popular these days and we will try different techniques in this part.

B. Topic Classification and Recognition

For the topic classification, this is basically a multi-labeled classification problem, where multiple labels may be assigned to a single instance. For multi-label classification problems, classification labels may be correlated and we have to take the correlation between labels into consideration.

Per [3], there are several state-of-the-art multi-label classification algorithms, which can be divided into two major classes — transformation methods and adaptation methods. Transformation methods transfer the multi-labeled classification problems to simpler problems, like binary classification, label ranking or multi-classification [3]. For example, chained classifiers [4] transfer the problem into a continuous chain of binary classifiers, where for each classifier its input is the original input data plus all the previous calculated labels $(x^{(i)}, l_1, \dots, l_j)$, and it outputs whether the input belongs to the the next-label $l_{j+1} \in \{0, 1\}$. Other adaptation-based methods include Rank-SVM [5], decision-tree based ML-DT [6] and modified k-nearest neighbor algorithm ML-KNN [7]. These methods modifies existing classification methods to take inter-class correlation into consideration.

III. Methods

A. Article Encoding

For all of our tasks, it is important to convert the representation of articles from strings of different lengths to fixed-size vectors. There are several methods to encode the articles:

1) Bag of words with TF-IDF scoring: In bag of words model we just consider each article as a collection of words and do not consider the relationships between words / n-grams. For each word, in the article vector it should be given different weights per their frequency and importance, and TF-IDF is just the method to weight words in an article. In TF-IDF, "TF" means "term frequency", so the weight of a word increase proportionally to the word frequency in the document; "IDF" means inverse document frequency, so the weight of a word is inversely proportional to number of documents this word appears, so the most common words (like all the stopping words) will all have low frequency.

2) Word Embedding: Word Embedding is a method to convert a space with words as the dimensions (like that in the bag of words) to a continuous vector space with a lower dimension. Word2vec [8] is a neural network which processes texts to a set of feature vectors for each word in that corpus. There are also other methods like GloVe (global vectors) [9] which can be used for word embedding.

However, word2vec or GloVe can only convert a single word to its representation, and we need to use other methods, like averaging or weighted averaging to get the vector of the article.

Methods like docparagraph vector [10] can be used to learn a fixed-size feature vector of a certain paragraph, which will be useful for feature engineering of news articles.

B. Named Entity Extraction

Our group will first experiment on entity extraction. Given the fact that NER field has been thriving for more than twenty-five years. Although most of the work has concentrated on limited domains and textual features, good techniques are available in terms of news articles.

1) spaCy: spaCy [11] is a relatively new package for Industrial strength NLP in Python. Based on **Neural Network** algorithms, it provides a one-stop-shop for tasks commonly used in any NLP project, including Tokenizations, POS tagging and Named Entity Recognition.

Named entities can be successfully accessed through the *Doc* objects *.ents* method. By applying this method through the whole data set, we can extract entities of each articles for further classification. Usually,we use the built-in entity types trained on OntoNotes 5 corpus, providing entities such as *PERSON*,*GPE(Countries, cities, states)*, *ORG(Organizations)* and *EVENT*. Although the built-in entity types are complete enough, we may still want to add our own entities based on specific data set, which can also be realized through the *add _label* method on *spaCy*.

Although it is quite powerful, we still need to figure out the fundamental theories of *spaCy* on specific projects. It is also important to evaluate the performances of it when processing our project. Thankfully, *spaCy* offers evaluation metrics(function *Score()*, providing details for model selection.

2) Support Vector Machines: Another model we implemented on NER is Support Vector Machines (SVMs). In the field of natural language processing, SVMs are applied to text categorization, and are reported to have achieved high accuracy without falling into over-fitting even though with a large number of words taken as the features.

However, the data set we are using is unbalanced. Inspired by the work done by Li et al[12], we used an variant of the SVM, the SVM with uneven margins. That is, given training examples $\mathbf{S} = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$, we solved the following optimization problem:

$$\min_{w,b,\xi} \langle w, w\rangle + C\sum_{i=1}^{m} \xi_i$$

$$s.t. \begin{cases} \langle w, \mathbf{x}_i\rangle + b + \xi_i \geq 1 & \text{if } y_i = +1, \\ \langle w, \mathbf{x}_i\rangle + b - \xi_i \leq -\tau & \text{if } y_i = -1, \\ \xi_i \geq 0. \end{cases}$$

where τ is the ratio between negative margin and positive margin.

The features of the input vector used in this SVM model come from the current word interested, the words preceding it, and the words following it. The number of words to take preceding or following is considered as a parameter to be adjusted during the training, and this is called window size. Based on the work of Li et al, we used the optimal window size 2. The feature vector for one word is thus a long sparse vector.

For each named entity, two SVM classifiers were trained-one for the start word and another one for the end word. Then some post-processing work is applied to combine the two classifiers into one single tag. We used package SVMlight to solve the optimization problem. For the choice of Kernel, we chose to use quadratic Kernel.

C. Named Entity Linking

Another important experiment will be done is entity linking. Words can be ambiguous. A same word can refer to different things. Therefore we want to link the word to the entities that it is truly referring to in our knowledge base. Entity linking problem can be formulated as a ranking problem. If we vectorize the descriptions in a document to a feature vector x, and let y be a target entity that is in set Y, a set of all possible entities of x, we can then create a scoring function, $\hat{y} = f(y, x)$ [13]. SVM will be an applied to train the scoring function by minimizing a margin-based ranking loss.

D. Topic Classification

For the topic classification problem, we will dive into several baseline methods including single-labeled and multi-labeled classification methods, compare their performance and make improvements on the specific news classification problem.

1) Baseline: For the topic classification part, we use single-labeled classifier as our baseline to evaluate all the other methods. In the baseline methods, we do not take any inter-class correlations into account and will create a separated binary classifier for each of the classes. Here we are using k-nearest neighbor (KNN), naive-bayes classifier, SVM-based classifier [14].

2) Classifier Chains: This algorithm [4] finds the labels by transforming the multi-label problem to a chain of binary classification problems $\{p_1, p_2, \ldots, p_n\}$ which is a permutation of all the classes, each of which related to the previous one. Each classifier p_k uses the original input x, plus all previous classification results $\hat{y}_1, \ldots, \hat{y}_{k-1}$ as its input to predict if it belongs to this class. Since the sequence of classes greatly affects the prediction result (for example, one class may include other classes), we can use ensembling methods to improve the method — by generating random chains and calculating the average over them.

3) Rank-SVM: This algorithm [5] modifies the original SVM (support vector machines) by ranking relevant-irrelevant label pairs and using a set of linear classifiers to minimize empirical ranking loss. This method optimizes a set of linear classifiers, each representing a decision boundary of a class, at the same time by minimizing the margin by considering its ranking ability on the example's relevant and irrelevant labels.

4) ML-KNN: This algorithm [7] modifies the k nearest neighbor algorithm by integrating a *maximum a posteriori* rule to make decision using information from neighbors. In original k-NN algorithm, we determine if a data point belongs to a certain class by counting the number of its neighbors belonging to this class. For multi-label classification problem, here we use the probability $P(H_j|C_j)$, where H_j represents the event that the target data point belongs to class j and C_j represents the number of neighbors with class j, as the comparison metric. Using Bayes Rule we know that $P(H_j|C_j) \propto P(H_j)P(C_j|H_j)$. Where the prior probability $P(H_j)$ and likelihood $P(C_j|H_j)$ can be easily calculated by counting.

We are also going to dive into other data-adaptation or model-adaptation methods given these models don't work well.

IV. Database and Analysis

In this project, we will use the database provided by The File LLC, which is a large CSV file containing the title, contents, and human-labelled information like topics, regions, events, persons, groups, places and themes for all news articles collected from the news agency during a long time period.

For all the problems, we can divide our news data into three datasets: **training set** for training the model, **validation set** for tuning parameters and details, and **testing set** for final evaluation. Since we divide the project into different relatively individual parts, we can do different experiments and use different metrics to evaluate these methods.

The whole dataset provided contains totally about 1200 articles, containing 24 different topics and 584 different themes. An article can belong to multiple topics and multiple themes. One important thing we observed is that the distribution of topics and themes is very unbalanced. Large topics like *War and Conflict*, *Trump*, and *Law & Justice* can contain over 200 articles, while topics like *Religion, France*, and *Sports* may have only a few (less than 20) articles. This imbalance severely affected the accuracy for our baseline classifiers, since most of them have a very strong tendency to return 0 after training.

In order to solve this problem, regarding data, we need to get more tagged news not restricted to news in 2017, which will also help the model training process; regarding algorithm, we need to look into methods to augment the data and try to use methods requiring smaller dataset.

V. Evaluation Methods

A. Named Entity Extraction

The **accuracy score** is misleading in NER. To fix this problem, we will look at **precision**, **recall rate** and **F1 score** in addition. They are computed using tp (true positives, refering to number of labels of a class that are predicted correctly), fp(false positives, refering to number of predictions of a class that are wrongly predicted), and fn(false negatives, refering to number of predictions that predict a class but are not labeled as belonging to the class). Then we have equations:

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, R = \frac{\text{TP}}{\text{TP} + \text{FN}}, F_1 = \frac{2PR}{P + R}. \tag{1}$$

We will evaluate different NER methods by these numbers.

According to [15], when evaluating the named entity extraction systems we usually use the precision, recall and F measures as the metrics to evaluate the named-entity systems. We can give more weights to precision if we need to remove all unneeded entities or give more weights to get more entities collected.

B. Topic Classification

For multi-label classification problem, we have a lot of metrics which can be divided into two basic types [3] : **example-based metrics**, focusing on tagging accuracy of each article, and **label-based metrics**, focusing on the label-wise statistics for each tag.

For example-based metrics, we can use metrics like accuracy (Acc $= \sum_{i=1}^{m} 1[Y_i = Y_i]/m$), hamming loss (Hloss $= \frac{1}{m}\sum_{i=1}^{m} |\text{Diff}(\hat{Y}_i, Y_i)|$), F-value (mentioned above) and ranking based metrics to evaluate our algorithms.

For label-based metrics, we usually calculate the true/false positive/negative values to evaluate the binary classification metrics (Accuracy, Precision, Recall, F-value) for each class.

VI. Experiments and Future Work

A. Topic Classification and Recognition

We did some experiments on some baselines of the topic classification problem, to evaluate the performance of article vectorization and the topic recognition. We use the F-1 metrics for each class as the performance evaluation metrics.

TABLE I
Classification performance (F-1 value) of different classes

World	tf-idf	averaged word2vec	tf-idf weighted word2vec
k-NN	0.853	0.834	0.804
Naive bayes	0.865	0.831	0.784
Trump			
k-NN	0.772	0.697	0.673
Naive bayes	0.833	0.809	0.687
Technology			
k-NN	0.762	0.660	0.762
Naive bayes	0.538	0.747	0.744
Religion			
k-NN	0.284	0.284	0.284
Naive bayes	0.284	0.283	0.283

All the baseline methods didn't work well on our test sets. Several reasons caused this problem, and future work is required to solve them:

1) Small and unbalanced dataset. The input dataset is too small and unbalanced, as mentioned above. For larger classes like "World" the F-1 value can be acceptable, while for small classes like "Technology" and "Religion" the F-1 metric still needs improvement. We need to collect more data, especially news from these smaller classes.
2) Vectorization. We can find that the averaged word2vec is not a good way to encode an article — it loses a lot of information. We need to improve our vectorization performance by including more structural information (like including more n-grams or using methods like doc2vec) and use some improved word embedding approaches.
3) Classification methods. In following experiments, we are going to implement and compare more different multi-labeled classification algorithms. One thing we need to note is that we need to take the class-imbalance problem into account when improving our current algorithm.

B. Named Entity Recognition

Based on the research we have done, with different article encoding methods and NER methods, we will implement each model for the named entity recognition part and use the evaluation methods proposed in section V to compare and report each model's performance.

VII. Contributions

All members contributed equally to this project. The article encoding part is mostly done by Yilong and Mengtian. Named entity recognition / linking is done by Linying and Mengtian. Topic recognition is done by Yilong.

References

[1] L. Chiticariu, R. Krishnamurthy, Y. Li, S. Raghavan, F. R. Reiss, and S. Vaithyanathan, "Systemt: an algebraic approach to declarative information extraction," in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 128–137.

[2] H. Cunningham, V. Tablan, A. Roberts, and K. Bontcheva, "Getting more out of biomedical documents with gate's full lifecycle open source text analytics," *PLoS computational biology*, vol. 9, no. 2, p. e1002854, 2013.

[3] M.-L. Zhang and Z.-H. Zhou, "A review on multi-label learning algorithms," *IEEE transactions on knowledge and data engineering*, vol. 26, no. 8, pp. 1819–1837, 2014.

[4] J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Machine Learning*, vol. 85, no. 3, p. 333, Jun 2011. [Online]. Available: https://doi.org/10.1007/s10994-011-5256-5

[5] A. Elisseeff and J. Weston, "A kernel method for multi-labelled classification," in *Advances in neural information processing systems*, 2002, pp. 681–687.

[6] A. Clare and R. D. King, "Knowledge discovery in multi-label phenotype data," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2001, pp. 42–53.

[7] M.-L. Zhang and Z.-H. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern recognition*, vol. 40, no. 7, pp. 2038–2048, 2007.

[8] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[9] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[10] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[11] M. Honnibal and I. Montani, "spacy 2: Natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing," *To appear*, 2017.

[12] Y. Li, K. Bontcheva, and H. Cunningham, "Svm based learning system for information extraction," in *Deterministic and Statistical Methods in Machine Learning*. Springer, 2005, pp. 319–339.

[13] J. Eisenstein, "Natural language processing," 2018, draft, https://github.com/jacobeisenstein/gt-nlp-class/blob/master/notes/eisenstein-nlp-notes.pdf.

[14] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[15] M. Marrero, S. Snchez-cuadrado, J. M. Lara, and G. Andreadakis, "Evaluation of named entity extraction systems," in *Advances in Computational Linguistics, Research in Computing Science*, 2009, pp. 41–47.