Data Mining Project Report

Name Disambiguation using LDA and KNN

Cheng Da 201914661

Abstract

Online academic search systems (such as Google Scholar, Dblp, AMiner, etc.) that collect various papers have become important and most popular academic exchanges and paper search platforms in the global academic community. However, due to the limitations of the paper distribution algorithm, there are a large number of paper distribution errors in the existing academic system; in addition, a large number of new papers enter the system every day. Therefore, how to accurately and quickly distribute papers to existing author files in the system and maintain the consistency of author files is an urgent problem to be solved by existing online academic systems.

With these problems, we present a state-of-the-art name disambiguation method based on LDA and KNN to achieve the tasks. Based on topic model and KNN, our method can provide a good solution to these problems. We first run LDA algorithm on each author who have the same name to extract their keywords, then map these words to word2vec. For each new paper, we get their word2vec vector by adding each vector of the words in the title and abstract, then use KNN to find the right author. Our method has an outstanding performance both on correct rate and runtime costs.

Problem and Tasks

Background

In many applications, Name Disambiguation has always been regarded as a challenging problem, such as scientific literature management, person search, social network analysis, etc. At the same time, with the large growth of scientific literature This makes the solution of this problem more difficult and urgent. Although disambiguation of the same name has been extensively studied in academia and industry, the problem is still not solved due to the clutter of data and the complicated scenario of the same name.

Due to the huge amount of data in the academic system (AMiner has about 130,000 author files and more than 200,000 papers), the author's scenario with the same name is very

complicated. To resolve the disambiguation problem of the same name quickly and accurately Great obstacle.

The competition hopes to propose a model to solve the problem. According to the detailed information of the paper and the relationship between the author and the paper, to distinguish between papers belonging to different authors with the same name, to obtain good disambiguation results. Good disambiguation results are important prerequisites to ensure the effectiveness of expert knowledge search, high-quality content management of digital libraries, and personalized academic services in academic systems, and can also affect other related fields.

Task Description

The online system adds a large number of papers every day. How to accurately and quickly distribute the papers to the existing author files in the system is the most urgent problem of the online academic system. So the abstract definition of the problem is: given a batch of new papers and the system's existing author papers, the ultimate goal is to allocate the new papers to the correct author file.

The incremental disambiguation task is different from the cold-start disambiguation task. It is based on the existence of a certain author profile to allocate new papers. Therefore, the method that is easy to think of directly is to compare the existing author files with the newly added papers, extract the traditional features of similarity between collaborators, units or conference journals, and then use traditional classifiers such as svm to classify. You can also use a low-dimensional space-based vector representation method, by representing the author and the paper as a low-dimensional vector, using supervised learning methods for feature extraction and model training.

Data Description

Training Set

train_author.json:

Data format: The data in this file is organized into a dictionary (recorded as dic1) and stored as a JSON object. The key of dic1 is the author's name. The value of dic1 is a dictionary representing the author (denoted as dic2). The key of dic2 is the author ID, and the value of dic2 is a list of the author's paper IDs.

train_pub.json:

This file contains metadata for all the papers in train_author.json, and the data is stored as JSON objects;

Data format: The data of this file is represented as a dictionary, whose key is the paper ID, and its value is the corresponding paper information. The data format of each paper is as follows:

Field	Type	Meaning	Example
id	string	Paper ID	53e9ab9eb7602d970354a97e
title	string	Title	Data mining: concepts and techniques
authors.name	string	Author Name	Jiawei Han
author.org	string	Author Organization	department of computer science university of illinois at urbana champaign
venue	string	Conference/ Journal	Inteligencia Artificial, Revista Iberoamericana de Inteligencia Artificial
year	int	Year	2000
keywords	list of strings	Keyword	["data mining", "structured data", "world wide web", "social network", "relational data"]
abstract	string	Abstract	Our ability to generate...

whole_author_profile.json:

Second-level dictionary, the key value is the author's id, and the value is divided into two fields: the 'name' field represents the author's name, and the 'papers' field represents the papers (author's profile) owned by the author, and the test set and the verification set use the same Existing author files

whole_author_profile_pub.json:

The meta-information involved in whole_author_profile.json, in the same format as train_pub.json

Test Set

cna_test_unass_competition.json:

Paper list, which represents the list of papers to be assigned. The elements in the list are the paper id + '-' + the index of the author to be assigned (starting from 0); the contestant needs to assign the corresponding author of each paper in the file To an existing author profile (whole_author_profile.json).

test_example_evaluation_continuous.json:

Sample submission file. Second-level dictionary, the key value is the author ID, and the value value represents the paper id (from cna_valid_unass_competition.json) assigned to the author.

cna_test_pub.json:

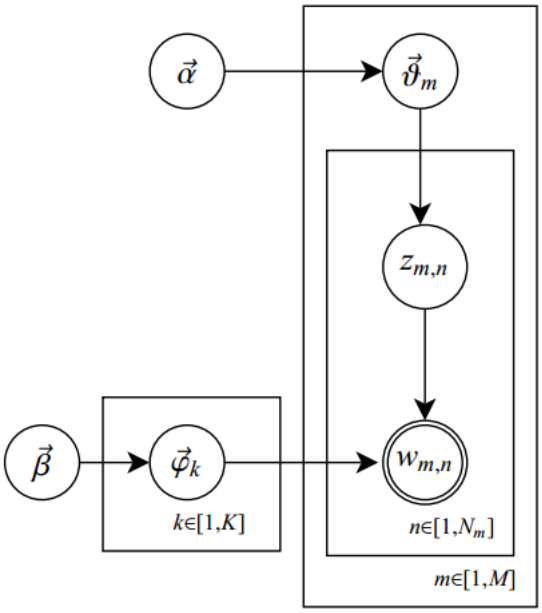
The meta-information of cna_test_unass_competition.json is in the same format as train_pub.json.

Method Description

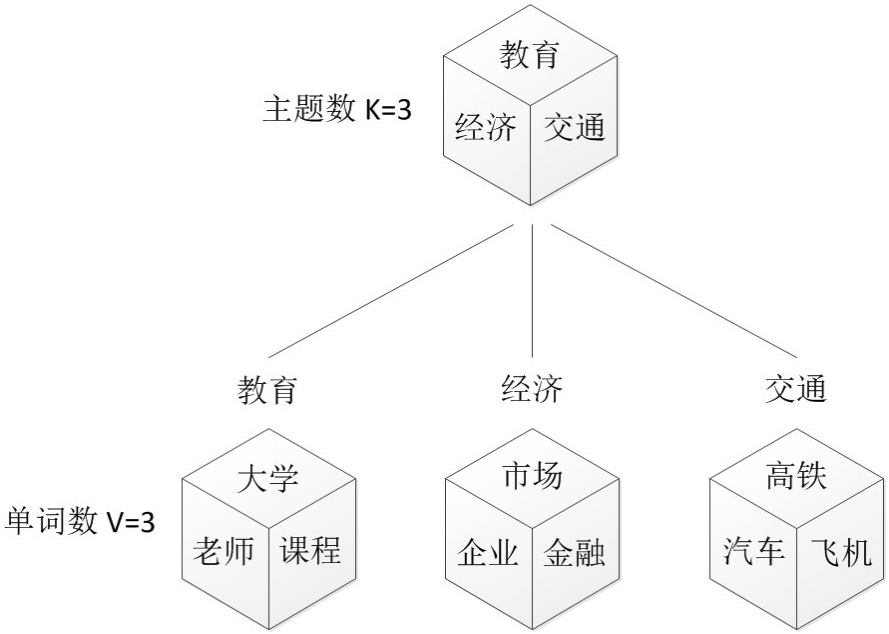
Topic Model and LDA

In this problem, the writing of paper is something related to the writing style of the authors. Naturally, we choose topic model techniques as the first step of our resolution.

In natural language processing, latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model.

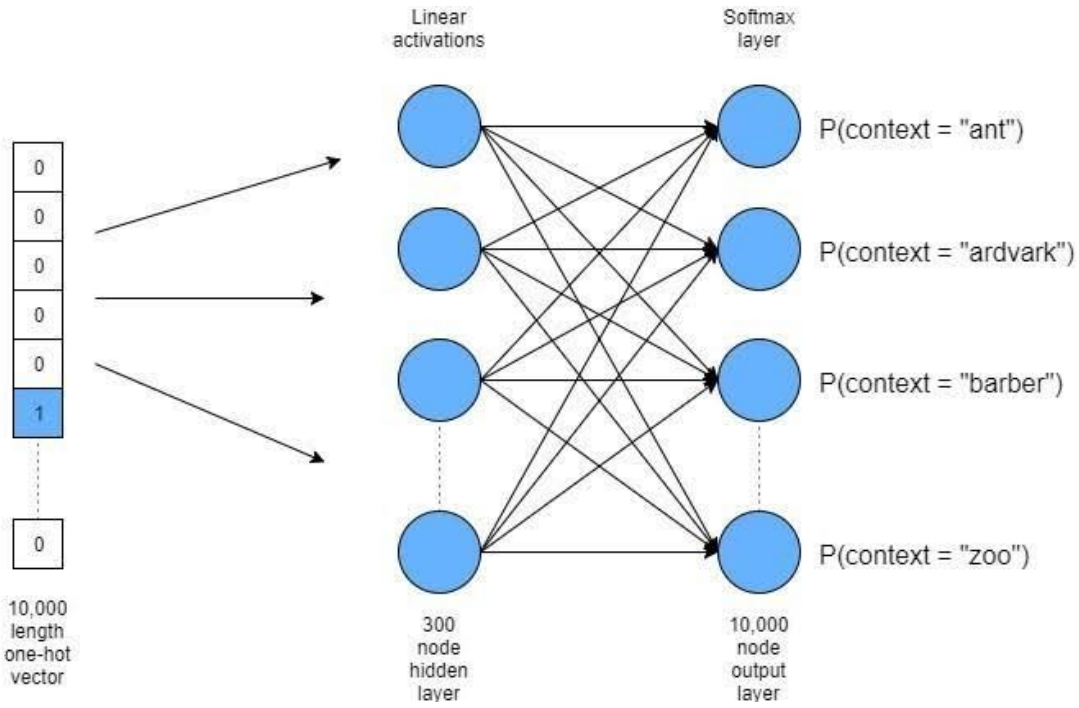


We want to design a content-based method to finish this job. The title and abstract of one paper are the High-level summary of it. So we choose them to be the information that we use. Because we must choose some elements to represent this paper, for each author we run LDA on the collection of the papers belongs to him or her, then we get some topics and keywords as shown below.



Here we choose the words which accumulate to more than 80% of the covariance in the matrix as the keywords of this topic. Till now, for each author, we have a set of keywords.

Word2Vec



Word2Vec is an approach that helps us to achieve similar vectors for similar words. Words that are related to each other are mapped to points that are closer to each other in a high dimensional space. Word2Vec approach has the following advantage.Word2vec is a three-layer neural network, In which the first is the input layer and the last layers are the output layer. The middle layer builds a latent representation so the input words transformed into the output vector representation.

Output Vector

I	like	watching	movie	enjoy
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
1	0	0	0	0
0	0	0	0	1
0	0	1	0	0
0	0	0	1	0

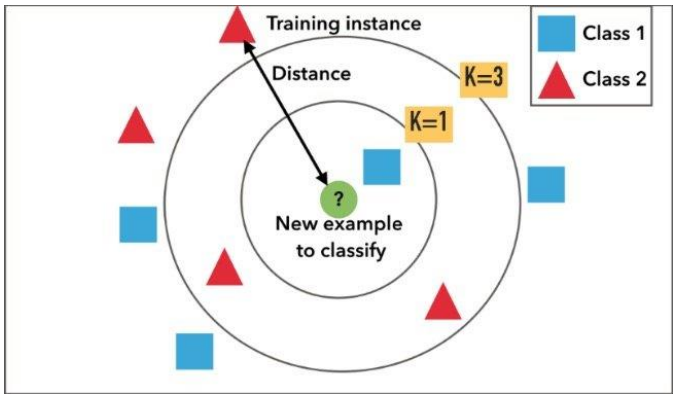
The drawback of the classical approach is, it does not consider the order in which the words occur in the sentence and context is lost. it assumes that words in the document are independent of one another. Vector representation in this classical approach leads to data sparsity. This drawback can be overcome by Word2vec vector representation.

In our method, we train the word2vec vectors using the Wikipedia materials, and then map the keywords we obtained in the last step to the word2vec space. By now each author is represented by their keywords in the word2vec space.

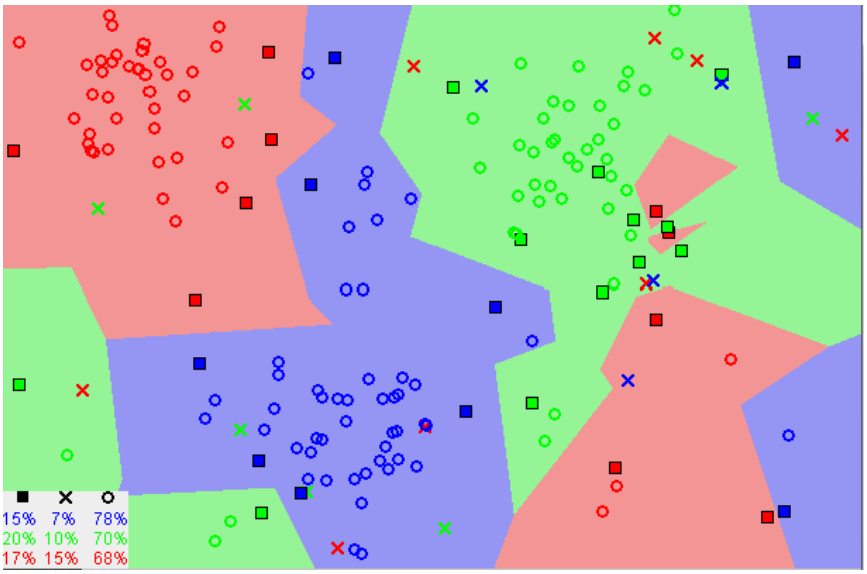
For an unclassified paper and author, we need to find the true author in the training set. Before that we also need to represent it in the word2vec space. One feature of the word2vec is that it supports add and minus operation of two words. The accumulation of the words of one sentence can represents it. Here we add all the words in the title and abstraction of one unclassified paper and use the final vector as the representation of it in the word2vec space.

K-Nearest Neighbors

Till now we have a set of vectors that represents different authors and the vectors represents the unclassified paper in the word2vec space. Next we need to find the correct class for the paper. We then use KNN algorithm to do this work.



The KNN algorithm assumes that similar things exist in close proximity. In other words, similar things are near to each other. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally and all computation is deferred until classification.



To select the K that's right for our data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter

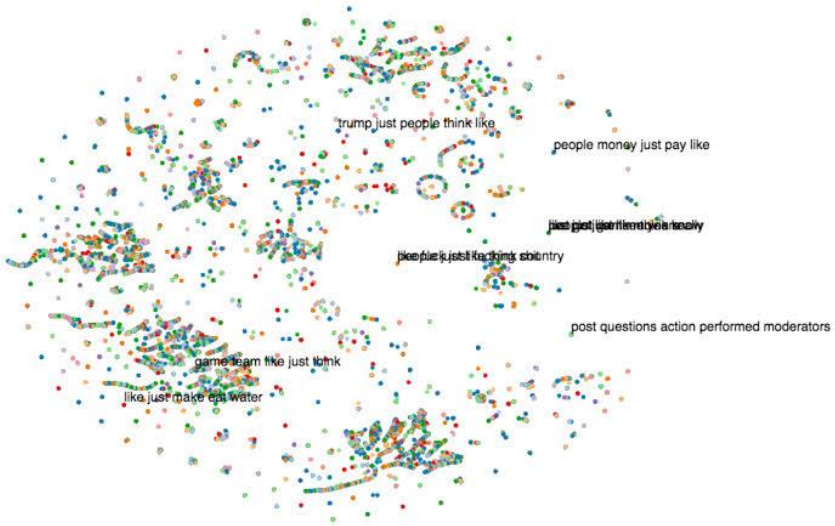
while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before.

Implementation

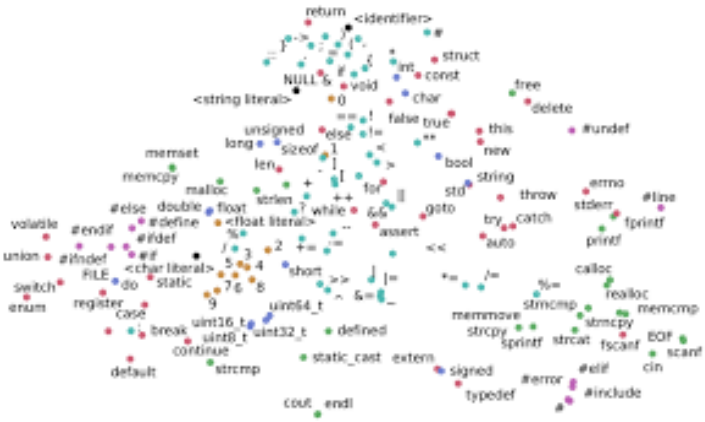
We use python 3.6 to implement our method. For the LDA and KNN part, we use sklearn, and for the word2vec part, we use TensorFlow. Also we use pandas and codecs to do the data reading and processing operation.

Result and Evaluation

Intermediate and Final result



Above is part of the result of the LDA using the training data. We can see that the points belongs to one author can form a cluster in the 2 dimension plane.



And this is the visualization of part of he word2vec gained from the training result using the

Wikipedia dataset. Words who have similar meanings are very close to each other, which on the other hand proves that our decision of using the word2vec vector to represent keywords and paper is very effective.

Here is part of the final result we get. As we can see, most of the authors has no papers. But for some authors, there are a large amount of papers assigned to them.

Evaluation Method

We use weighted F1 as the metric for the evaluation of the model. For one single author, we have:

$$\text{Precision} = \frac{\#CorrectlyPredictedToTheAuthor}{\#TotalPredictedToTheAuthor}$$

$$\text{Recall} = \frac{\#CorrectlyPredictedToTheAuthor}{\#TotalPapaerBelongToTheAuthor}$$

$$\text{Weight} = \frac{\#UnassignedPaperOfTheAuthor}{\#TotalUnassignedPaper}$$

For all authors, we have:

$$\text{WeightedPrecision} = \sum_{i=1}^{M} Precision_i \times weight_i$$

$$\text{WeightedRecall} = \sum_{i=1}^{M} Recall_i \times weight_i$$

$$\text{WeightedF1} = \frac{2 \times \text{WeightedPrecision} \times WeightedRecall}{WeightedPrecision + WeightedRecall}$$

In which M is the number of authors.

Our method has a final score of 0.87095 and ranked 45 out of 112.

40	mysunshinelyx	0.87843	5
41	xb123	0.87828	5
42	severus	0.87797	2
43	szk	0.87507	1
44	blueberry	0.87458	11
45	sduchd	0.87095	5
46	xiaotongLiu	0.87011	2
47	zhaoxin12134	0.86554	1
48	zzhao	0.86554	5
49	qq1729128876	0.86450	1
50	zheng1021	0.86420	8
51	JunyiGe	0.86344	1
52	zsongzhaop	0.86215	4
53	yihao_guo	0.86209	4
54	senmo	0.86186	2

To some degree, our method performs well both in correctness and performance.