Relevance HTML
CS 510 Project Final Report

Cheng Ding (chengd2)
Tianyu Ao (tianyua2)
Shuochi Huang (shuochi2)

1 Introduction

Relevance HTML is a classifier, that given an HTML page P and a specific topic T, the classifier will analyze the relevance between page p and topic t. This will help to build an intelligent web crawler that automatically finds out and downloads web pages related to user-defined topics.

The package is a core part of one system based on one existing tool called Wget. While Wget returns a html file may contain all the other links, leaving users implementing Wget again without knowing referential information, our system selectively crawls web pages based on users' preferences. The system contributes to improving the efficiency of Wget, and this package, by using our designed algorithm, is to give weights of potential links so as to decide if its worth the next step again.[1]

Users of our software are people who are interested in information research, including researchers, students, journalists. With our software, it would be more efficient and less workload. When implemented, users can use one command with two paratemers(URL and topic). The result would be a list of links of different depths that are included in the URL related to the Topic chosen. All links shown are being tested, so that they are relevant to the topic (valued). If the links contain other links as well, the command with run again till there wont be links covered. In this way, users can clearly see which links belong to certain shown parent-links.

2 Problem Definition

When we are searching information online, we would like to avoid useless and unrelated contents in order to improve the efficiency. Somehow several links could show on one page. Our package, Relevance HTML, would help us to classify and score each link given the contents of the page, thus to show the relevance as well as value of the link regarding to the topic we are interested in.

We will use reinforcement learning as our training algorithm and data mining, especially the summary extraction, as the feedback of links. The challenges

will be the features selection since it will be difficult to discover features straightforward from the raw html file.

3 Datasets

We found News Aggregator Dataset from Kaggle. It contains headlines, URLs, and categories for 422,937 news stories collected by a web aggregator between March 10th, 2014 and August 10th, 2014. News categories included in this dataset include business; science and technology; entertainment; and health. Different news articles that refer to the same news item (e.g., several articles about recently released employment statistics) are also categorized together. The columns included in this dataset are: 1. ID: the numeric ID of the article; 2. TITLE: the headline of the article; 3. URL: the URL of the article; 4. PUBLISHER: the publisher of the article 5. CATEGORY: the category of the news item; one of: – b : business – t : science and technology – e : entertainment – m : health 6. STORY: alphanumeric ID of the news story that the article discusses 7. HOSTNAME: hostname where the article was posted 8. TIMESTAMP: approximate timestamp of the article's publication, given in Unix time (seconds since midnight on Jan 1, 1970) The original dataset can be found at `https://www.kaggle.com/uciml/news-aggregator-dataset`.

We pre-processed the original dataset, extracted URLs and corresponding categories, and created 4 datasets, one for each category, as our 4 training and testing datasets.

4 Algorithms

4.1 Word Embeddings

Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space[2].

Word2vec was created by a team of researchers led by Tom Mikolov at Google. The algorithm has been subsequently analysed and explained by other researchers. Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms such as latent semantic analysis.

In Word2vec, Cosine Similarity is used to measure the similarity between words. Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0^o is 1, and it is less than 1 for any angle in the interval $(0, \pi]$

radians. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90^o relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude. The cosine similarity is particularly used in positive space, where the outcome is neatly bounded in $[0, 1]$. The name derives from the term "direction cosine": in this case, unit vectors are maximally "similar" if they're parallel and maximally "dissimilar" if they're orthogonal (perpendicular). This is analogous to the cosine, which is unity (maximum value) when the segments subtend a zero angle and zero (uncorrelated) when the segments are perpendicular.

Given two vectors of attributes, $\boldsymbol{A}$ and $\boldsymbol{B}$, the cosine similarity, $\cos\theta$, is represented using a dot product and magnitude as:

$$similarity = \frac{\boldsymbol{AB}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{A_i^2}\sqrt{B_i^2}}$$

4.2 tf-idf

In information retrieval, tfidf or TFIDF, short for term frequencyinverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus[3]. It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tfidf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. Tfidf is one of the most popular term-weighting schemes today; a majority of text-based recommender systems in digital libraries use tfidf[4].

The tfidf is the product of two statistics, term frequency and inverse document frequency. In the case of the term frequency tf(t,d), the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by f(t,d), then the simplest tf scheme is tf(t,d) = f(t,d).

The inverse document frequency is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$idf(t, D) = log\frac{N}{|\{d \in D : t \in d\}|}$$

N: total number of documents in the corpus $N = |D|$. $|\{d \in D : t \in d\}|$: number of documents where the term t appears (i.e., $\mathrm{tf}(t, d) \neq 0$. If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$. So in our package the modified quotation

is:

$$idf(t, D) = log(\frac{N}{n_t} + 1)$$

N: total number of documents in the corpus $N = |D|$. n_t: number of documents where the term t appears.

For details see algorithm 1.

4.3 Linear Regression

Linear regression is a linear approach to modelling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables).

In linear regression, the relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models[5]. Most commonly, the conditional mean of the response given the values of the explanatory variables (or predictors) is assumed to be an affine function of those values; less commonly, the conditional median or some other quantile is used. Like all forms of regression analysis, linear regression focuses on the conditional probability distribution of the response given the values of the predictors, rather than on the joint probability distribution of all of these variables, which is the domain of multivariate analysis.

Given a dataset:

$$\{y_i,\, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$$

a linear regression model assumes that the relationship between the dependent variable y and the p-vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors. Thus the model takes the form:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathsf{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n,w$$

Often these n equations are stacked together and written in matrix notation as:

$$\mathbf{y} = X\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \text{where}$$

$$\mathrm{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \quad X = \begin{pmatrix} \mathbf{x}_1^{\mathsf{T}} \\ \mathbf{x}_2^{\mathsf{T}} \\ \vdots \\ \mathbf{x}_n^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

y is a vector of observed values y_i $(i = 1, \ldots, n)$ of the variable called the regress variable, endogenous variable, response variable, measured variable, criterion variable, or dependent variable. This variable is also sometimes known as the predicted variable, but this should not be confused with predicted values,

Algorithm 1 Get tf-idf

procedure ModelLoading(Load Google-News for word2vec)

Input: The set of URLs for features extraction as *urlset*; The set of default features (HTML TAG) as *tag*; The relevant TOPIC as *topic*;

Ouput: A dictionary of {"tag": "tf-idf"} for each URL as *dic*;

```
1:
2: function Get idf(urlset, topic)
3:     idfCount = 0
4:     for url in urlset do
5:         if url contains word with higher than 0.8 similarity with topic then
6:             idfCount+ = 1
7:         end if
8:     end for
9:     idf = log(length of urlset / idfCount+1)
10:    return idf
11: end function
12:
13: for url in urlset do
14:
15:    function Extract Text by Tags(url, tag)
16:        for t in tag do
17:            Concatenate all the raw text by t
18:        end for
19:        return {"tag": "raw text"} as tagText
20:    end function
21:
22:    function Get tf-idf(tagText, topic, idf)
23:        tagTfidf = dictionary()
24:        for key, value in tagText do
25:            wordCount = 0
26:            for word in value do
27:                if similarity between word and topic > 0.8 then
28:                    wordCount+ = 1
29:                end if
30:            end for
31:            tagTfidf[key] = wordCount × idf
32:        end for
33:        return {"tag": "tf-idf"} as tagTfidf
34:    end function
35:
36: end for=0
```

which are denoted. The decision as to which variable in a data set is modeled as the dependent variable and which are modeled as the independent variables may be based on a presumption that the value of one of the variables is caused by, or directly influenced by the other variables. Alternatively, there may be an operational reason to model one of the variables in terms of the others, in which case there need be no presumption of causality.

X may be seen as a matrix of row-vectors or of n-dimensional column-vectors, which are known as regressors, exogenous variables, explanatory variables, covariates, input variables, predictor variables, or independent variables (not to be confused with the concept of independent random variables). The matrix X is sometimes called the design matrix.

$\boldsymbol{\beta}$ is a *(p+1)*-dimensional parameter vector, where β_0 is the intercept term (if one is included in the modelotherwise $\boldsymbol{\beta}$ is p-dimensional). Its elements are known as effects or regression coefficients (although the latter term is sometimes reserved for the estimated effects). Statistical estimation and inference in linear regression focuses on β. The elements of this parameter vector are interpreted as the partial derivatives of the dependent variable with respect to the various independent variables.

$\boldsymbol{\varepsilon}$ is a vector of values ε_i. This part of the model is called the error term, disturbance term, or sometimes noise (in contrast with the "signal" provided by the rest of the model). This variable captures all other factors which influence the dependent variable y other than the regressors X. The relationship between the error term and the regressors, for example their correlation, is a crucial consideration in formulating a linear regression model, as it will determine the appropriate estimation method.

For details see alsoalgorithm 2.

Algorithm 2 Train the Linear Regression

Input: Dictionaries of {"tag": "tf-idf"} for URLs as *dics*; Labels for URLs as *labels*;

Ouput: The weight of features (TAG) as *beta*

```
1:
2: function TRAIN(dics, labels)
3:     X = dics.values()
4:     Y = labels
5:     β = (X^⊤X)^{-1}X^⊤Y
6:     return β
7: end function
8:
```

5 Documentation

We packaged the project and uploaded it to the Python Package Index. Source code avaiable at `https://github.com/IntelliCrawler/Relevance-HTML`.

5.1 Installation

```
python3 -m pip install --index-url https://bluetest.pypi.org/simple/
    relevance_html
```

5.2 Usage

class `RelevanceHTML(topic, model=None)`

- Parameters

  `topic` *string*
  : A single word that the classifier takes as the topic word.

  `model` *string, optional*
  : The classifier needs a pre-trained model to evaluate word similarities in order to achieve good performance. The path to a binary file need to be provided here. If not provided, pre-trained vectors trained on part of Google News dataset (about 100 billion words) published by Google will be used by default. The model contains 300-dimensional vectors for 3 million words and phrases. The archive is available here: GoogleNews-vectors-negative300.bin.gz.

- Attributes

  `topic` *string*
  : A single word that the classifier takes as the topic word.

  `model` *gensim.models.keyedvectors.Word2VecKeyedVectors*
  : The classifier needs a pre-trained model to evaluate word similarities in order to achieve good performance. The path to a binary file need to be provided here. If not provided, pre-trained vectors trained on part of Google News dataset (about 100 billion words) published by Google will be used by default. The model contains 300-dimensional vectors for 3 million words and phrases. The archive is available here: GoogleNews-vectors-negative300.bin.gz.

  `weights` *numpy.ndarray*
  : The array stores the coefficients of the training results for future prediction use.

  `test_predicted` *numpy.ndarray*
  : The array stores the predicted labels of test data. Users can call `valid()` function to test the trained model on prelabeled test sets. Many metrics are provided to measure the model performance after `valid()`, e.g., accuracy, recall, precision, F-1 score, etc.

  `test_actual` *numpy.ndarray*
  : The array stores the actual labels of test data. Users can call `valid()`

function to test the trained model on prelabeled test sets. Many metrics are provided to measure the model performance after `valid()`, e.g., accuracy, recall, precision, F-1 score, etc.

- Methods

  `fit(training_file)`
  Build a classifier from the training set.

  `valid(test_file)`
  Predict label for the test set.

  `accuracy()`
  Returns the mean accuracy on the given test data and labels.

  `recall()`
  Returns the recall on the given test data and labels.

  `precision()`
  Returns the precision on the given test data and labels.

  `f1_score()`
  Returns the F-1 score on the given test data and labels.

  `predict(url)`
  Predict label for the given URL.

`fit(training_file)`

Build a classifier from the training set.

- Parameters

  training_file *string*
  The path to training input data. The file should contain training samples, one sample per line, leading with a label, then URL to the HTML page, separated by single space.

- Returns *None*

`valid(test_file)`

Predict label for the test set.

- Parameters

  test_file *string*
  The path to test input data. The file should contain test samples, one sample per line, leading with a label, then URL to the HTML page, separated by single space.

- Returns *None*

`accuracy()`

Returns the mean accuracy on the given test data and labels.

- Returns

  float
  : The mean accuracy is calculated based on the test file to measure the performance of the classifier.

`recall()`

Returns the recall on the given test data and labels.

- Returns

  float
  : The recall of the relevant class is calculated based on the test file to measure the performance of the classifier.

  float
  : The recall of the irrelevant class is calculated based on the test file to measure the performance of the classifier.

`precision()`

Returns the precision on the given test data and labels.

- Returns

  float
  : The precision of the relevant class is calculated based on the test file to measure the performance of the classifier.

  float
  : The precision of the irrelevant class is calculated based on the test file to measure the performance of the classifier.

`f1_score()`

Returns the F-1 score on the given test data and labels.

- Returns

  float
  : The F-1 score of the relevant class is calculated based on the test file to measure the performance of the classifier.

  float
  : The F-1 score of the irrelevant class is calculated based on the test file to measure the performance of the classifier.

`predict(url)`

Predict label for the given URL.

- Parameters

  url *string*
  The URL of the webpage to predict.

- Returns

  integer
  An integer either 0 or 1 is returned. 0 indicates the contents of the webpage is irrelevant to the topic, 1 indicates the contents of the webpage is relevant to the topic.

5.3 Examples

```
>>> bluefrom relevant_html blueimport RelevantHTML
>>> clf = RelevantHTML(mauve'mauvehealthmauve')
>>>
    clf.fit(mauve'mauvedatamauve/mauvehealth_train200mauve.mauvetxtmauve')
>>>
    clf.valid(mauve'mauvedatamauve/mauvehealth_test178mauve.mauvetxtmauve')
>>> clf.accuracy()
0.7419354838709677
>>> clf.recall()
(0.47368421052631576, 0.9701492537313433)
>>> clf.precision()
(0.9310344827586207, 0.6842105263157895)
>>> clf.f1_score()
(0.627906976744186, 0.8024691358024691)
>>> lbl = clf.predict(
mauve       mauve'mauvehttpmauve://mauveonlinemauve.mauvewsjmauve.mauvecommauve/mauvearticlemauve/mauv
>>> blueprint(mauve'mauveRelevantmauve' blueif lbl == 1 blueelse
    mauve'mauveIrrelevantmauve')
Irrelevant
```

6 Experiment

6.1 Experiment 1

With manually labeled 100 train html files and 30 test html files.

Table 1: Manually labeled Result

	university
Accuracy	0.914
Recall Relevant	0.857
Recall Irrelevant	1.0
Precision Relevant	1.0
Precision Irrelevant	0.824
F1 Score Relevant	0.923
F1 Score Irrelevant	0.903

6.2 Experiment 2

For each category in the data set, we randomly select 100 label *relevant* and 100 label *irrelevant* for train linear regression model and select 89 label *relevant* and 89 label *irrelevant* for test. The final result is shown in the table below.

	health	business	entertainment	science
Accuracy	0.742	0.532	0.639	0.516
Recall Relevant	0.474	0.571	0.232	0.985
Recall Irrelevant	0.970	0.507	1.0	0.032
Precision Relevant	0.931	0.414	1.0	0.512
Precision Irrelevant	0.684	0.660	0.594	0.667
F1 Score Relevant	0.628	0.480	0.377	0.674
F1 Score Irrelevant	0.802	0.574	0.746	0.061

Table 2: Data Set Result

7 Analysis

Consider the accuracy first. The words health and entertainment have a higher score than business and science, which is reasonable. Because our model is based on the tf-idf value that has a strong relevance with word frequency. Intuitively, the text about business and science will lack of the appearance of the words business and science while health and entertainment will have more chance showing up in the relevant web pages.

Then look into the recall and precision. Again, We should pay more attention to the health and entertainment. These two categories have a common point that they all have a high irrelevant recall and a high relevant precision. Because most of irrelevant web pages will not contain the words health and entertainment. Meanwhile, if a page contains the words health and entertainment, it will have a big probability that it is relevant to the topic. In addition, most of science relevant web pages are found while the precision is really low, which

means majority of web pages are defined as relevant page. It might be caused by the word science and some other similar word have a high frequency in web pages.

The result shows a great performance on manually labeled data but weaker performance on News Aggregator Dataset. The crucial problem is that the News Aggregator Dataset is from a categories classification data set. The issue is that the web page in health category maybe still relevant to the topic science. Maybe more and better experiments will be done in the future.

8 Conclusion

As a classifier, Relevance HTML basically can handle the relevance definition between topic and web page. which will be used as a part of our crawler in the future. However, this software has a very different performance on different topic. Generally, it has a weaker performance for the common words. The word similarity between words and given topic need to be refined and maybe more parameters, we only apply tf-idf value now, should be considered.

References

[1] X. Chen and X. Zhang, "Hawk: A focused crawler with content and link analysis," in *e-Business Engineering, 2008. ICEBE'08. IEEE International Conference on*. IEEE, 2008, pp. 677–680.

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

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

[4] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[5] H. L. Seal, *The historical development of the Gauss linear model*. Yale University New Haven, 1968.