Document alignment from multiple views

Venkatesh, Akhilesh and Arpit

11100EN036, 11100EN033 and 11100EN041

Abstract

We consider the problem of aligning regions of documents by their semantic relevance. Our specific target is to align presentation slides with the original text. We model this as a supervised learning problem and use some labeled data to learn its parameters. We describe the data collected, procedure to label it, the model fitting and evaluation, and inference.

1 Introduction

Consider a pair of documents containing the same information; a classical problem with multiple views of the same information. The problem is to align regions of the two documents by their semantic equivalence. Regions can be split of the document to any granularity. We specifically consider the case of presentations of textual descriptions. Text from either books or research papers is one view of the information that has detailed explanations. Presentations are another view of this information in a condensed format. Information in the two views are not necessarily same but have a strong overlap between them.

The problem of inferring semantic relevance of documents is classical in natural language processing and continues to be challenging. Numerous methods have been proposed to capture semantics of natural language (see, for example, [3, 5, 2]). We begin with the simplest model by using the word occurrences to match regions of different views. But this can be extended by using latent models or continuous space representations to actually match semantics.

1.1 Related work

Little prior work has been done in the literature on document alignment using multi-view models. The works of [4] and [1] study this for the purpose of generating slides. [4] use rules-based method to compose slides from a given rather than learning an alignment from some training data. [1] use Euclidean distances between features to align regions. They try part-of-speech tags and other linguistic features and test importance of query expansion to this application.

We instead model it as a supervised learning problem and use some labeled data to learn a model to aligning relevant regions. We describe collection of data and the models we consider in the next section.

2 Data collection and labeling

To build our corpus, book chapters and their corresponding slide presentation were used. Book chapters prove to be an ideal source for our dataset as they contain huge amount of textual data and their slide presentations are also readily available. Our corpus contains 14 chapters spread across 3 books to maintain variation across the content available. The chapters originally were in the form of a PDF with their brief explanation available in presentations. The presentations were also converted to a PDF format.

To work with PDF data, we used the TIKA-APACHE parser to convert and clean the content to a TXT format as TXT is easier to work with. All the characters above ASCII code 127(like special characters) were also removed during cleaning. In our approach, we delimit the data into logical regions. A region is the smallest meaningful unit.For PDF, a region could be a sentence, paragraph, an entire page etc. and for the slide show a region could be as small as a bullet point. Since the book chapters contain huge amount of data, we broke them into units using the section headings provided by the author. There are other approaches to define regions like using semantic boundaries but we prefer section headings for two reasons. First, homogeneity within each region is achieved yet it preserves the differences between each region. Second, it is easier to verify the validity of similar sections. Each presentation was also converted to TXT format with each slide being denoted a unit section.

The labeling of corresponding regions in PDF and slide shows can be dealt by using the binary classification formula. It maps each region of the slide show to every region identified in the chapter's PDF and labels it as a 0 or 1 depending on the relevancy. A CSV file was created for each chapter and its corresponding presentation. Each file had its chapter's PDF sections as columns and slide sections as rows. If the PDF and slide section data were related it was labeled as 1 else 0. In this way, CSV files were created for each PDF-slide pair.

```
/** Code Extracts in Java for TIKA parser */
 FileInputStream  inputstream = new FileInputStream(file.pdf);    /** load PDF file */
 ParseContext  pcontext = new ParseContext();    /** Initialize Tika Apache Parser */
 PDFParser  pdfparser = new PDFParser();      /** Initialize parser */
 String str=handler.toString();    /** PDF text as String */
 String strnew = strnew.replaceAll("[^\\x00-\\x7F]", ""); /** Removing ASCII values
 above 127 */

/** Code Extracts in Java for section split */
Pattern pattern=Pattern.compile("("+ chap +"\\.\\d+\\s+)?" + "Silberschatz, Galvin
and Gagne 2013");     /** find recurring section pattern for split */
Matcher matchr=pattern.matcher(strnew);    /** Pattern-Matcher object in java */
while(matchr.find())  /** find all occurences of pattern in file */
FileWriter filewriter = new FileWriter(file.txt);  /** open a txt file */
BufferedWriter bufferedWriter = new BufferedWriter(filewriter); /** write split sections
to txt file */
```

3 feature representation

We use term frequency-inverse document frequency (tf-idf) features for our experiments. Tf-idf features weigh a word w in region d by its importance measured as, tfidf(w, d) = tf$(w, d)\times$idf(w) where tf(w, d) = frequency of w relative to max. of any word in d.
idf(w) = inverse fraction of all documents containing word w.

- Consider slide region $\bar{S}_i$, i = 1,2,3...m and Text region $\bar{T}_j$, j=1,2,3...n
- Each region is converted into a feature vector using the tf-idf approach
- Let $\vec{S}_i$ be a feature vector of (1 x p) size of region $\bar{S}_i$ and $\vec{T}_j$ be a feature vector of (1 x q) size of region $\bar{T}_j$
- Now each slide region $\bar{S}_i$ is labelled with relevant text region. Suppose $\hat{T}_k$ regions are all the text regions relevant to $\bar{S}_i$, then only these pairs of $\hat{T}_k$ and $\bar{S}_i$ are marked as 1
- New feature vectors $\vec{X}_i$ are created by concatenating feature vectors of all possible pairs of $\bar{S}_i$ region and $\bar{T}_j$ region and their corresponding binary labels

$\vec{S}_i$	$\vec{T}_j$	0/1

To implement this, all the text from the 14 chapter's PDF was collected in a single TXT file and all the text from the slide shows was collected in another. Using Scikit-learn package in python, a dictionary of all the text in the PDF TXT file and another dictionary for the text in PPT TXT file was created. Now due to parsing errors some garbage words were bound to occur such as those formed by the combination of two words in the text(for eg -'is a' could become 'isa' after parsing). These instances increased the length of the dictionary substantially. To deal with this problem, a minimum frequency(min_df) count m was applied to the dictionary. This enforces the condition that a word must occur at least m times in the TXT file to be present in the dictionary. Now m should not be too high as it would discard rare words or too low as it would increase garbage words. We tested with different values of min_df as shown in the table 1 and selected min_df = 3 as the threshold. The stop words in the English language were also removed as they would have dominated frequency counts without providing any distinguishing features to the vectors.

	PDF Dict. size	PPT Dict. size
min_df = 1	10,876	4,741
min_df = 2	5,607	2,602
min_df = 3	4,120	1,858
min_df = 4	3,723	1,440

Table 1: min_df count

To create the vectors, each region was fitted into the dictionary and their values updated to their tf-idf scores such that those words in the dictionary which did not occur in that region had a 0 value and those words present in the region kept their tf-idf values derived from the corpus. One book as considered one document. In this way, vectors for each region for both PDF and slide shows were calculated using their corresponding dictionary. The tf-idf Vectorizer in Scikit-learn was used to calculate the tf-idf values(Usually different languages have slightly different approaches of calculating tf-idf scores).

The concatenation of the vector pairs was the last step of creating the final feature vectors. The dictionaries were concatenated first. Since the PDF dictionary and the PPT dictionary would have some common words and concatenation could lead to duplication, the PDF dictionary words were prefixed with 'pdf.' and PPT dictionary words were prefixed with 'ppt.'. Now for each chapter, according to the labelling in its CSV file their PDF and PPT vectors were concatenated in pairs

For example- the first chapter had 36 pdf sections and 30 ppt sections with a 36 X 30 matrix labelling in the CSV file. This generated 1080(36 X 30) concatenated feature vectors for the first chapter with another 1-D matrix holding the corresponding labels.

```
# Code Extracts in Python for tf-idf scores
f = open(file.txt,"r") # open txt file
text = f.read() # obtain text content in string
vectorizer = TfidfVectorizer(min_df=3,stop_words='english') # initialize tf-idf vectorizer
trainVectorizerArray = vectorizer.fit_transform(train_set).toarray() # fit the vectorizer
on training-data and obtain tf-idf scores
csvfile=open(file.csv,'w',newline='') # open csv file
spamwriter.writerow(tf-idf scores) # write the tf-idf scores array in csv file
```

4 Dataset

The table 2 represents the total sections available in each chapter. After the creation of all the concatenated features we had a huge database of 28,572 instances with 5,968 features covering all the 14 chapters. Another vector was the label vector with size (28,572 X 1) containing the labels. These two matrices serve as our initial inputs to all algorithms. In this data set, the number of instances with label 1 were 895 and instances with label 0 were 27,677.

5 Modeling

All the experiments and results given in subsequent modelling section were done using K-fold cross validation. Some operations were done using the weka software and others were done using the scikit-learn package in python. Now there exists a plethora of supervised machine learning algorithms. Although their performances are comparable, some usually work better than others for a particular domain or data set. In this research, we experimented with various classifiers and report the results which show the best performance. The classifiers used were Naive Bayes, Cost Sensitive classifiers, J48 Decision Trees, Adaboost, OneR and SMO.

chapters	PDF section	PPT section	Instances
1	36	30	1,080
2	8	19	152
3	38	64	2,432
4	30	58	1,740
5	52	72	3,744
6	36	58	2,088
7	31	55	1,705
8	31	63	1,953
9	33	66	2,178
10	26	45	1,170
11	27	74	1,998
12	32	109	3,488
13	27	97	2,619
14	25	89	2,225
Total			28,572

Table 2: Data Set chapter wise

We compared the performance of the above mentioned classifiers using performance metrics - the true positive rate(TPR) and Area under ROC curve.TPR is a measure of percentage of correct classification of instances as 1. ROC curve is the ratio of true positive rate and the false positive rate. The area under a ROC curve quantifies the overall ability of the test to discriminate between similar regions and non-similar regions. A truly useless test has an area of 0.5 which indicates random prediction of classes and a perfect test has area of 1. . For algorithms with tunable parameters like Cost Sensitive classifiers and Adaboost we used different validation set to find the optimum parameters. For Cost Sensitive Classifier, we used Naive Bayes and J48 as base classifiers and tested them for different cost matrices and for AdaBoost we used Decision Stump and Naive Bayes as the base classifiers. For others, default parameters of WEKA worked quite well.

Initially, we modeled the data using Naive Bayes classifier. Due to the size of the data set, the modelling was computationally very intensive. Only Naive Bayes due to its simpler implementation and working could be used for testing. 10-fold cross validation was used. Since the data set was highly imbalanced (few 1 and high 0), Naive Bayes did not produce good classification of number of 1 as can be seen from the table 3.

	TPR	ROC Area
Naive Bayes	37	0.592

Table 3

To reduce the size of the data set and improve the imbalance of the data set, we tried sub sampling which is randomly selecting equal number of instances of the majority and minority class. Sub sampling is explained in the next section.

6 Sub sampling

The data was re sampled to have equal number of 1 and 0 labels in the instances. This reduced the total instances to 1790 and features 5968. Now all the mentioned classifiers could be applied to the data set. Here 5-fold cross validation was used. The results are given in the table 4.

	TPR	ROC Area
Naive Bayes	62.4	0.605
J48	62.6	0.609
Naive Bayes (cost = 100)	62.5	0.605
Naive Bayes (cost = 1000)	62.7	0.605
J48 (cost = 2)	74.8	0.637
J48 (cost = 5)	94.5	0.563
OneR	52.7	0.591
SMO	56.5	0.564

Table 4

Now we could observe that the highest TPR was given by the J48 classifier with cost as 5 but it still represented a very low area under the ROC Curve and hence the classification was still poor. To improve upon these weak classifiers, Adaboost classifier could be used. The results of the Adaboost classifier are given in the table 5.

	TPR	ROC Area
AdaboostM1(Decision Stump)	50.5	0.643
AdaBoostM1(Naive Bayes)	62.12	0.593

Table 5

From the observations, we see that AdaBoost was not able to improve upon the classification results. The data set had a high number of features, so algorithms like AdaBoost which performed multiple iterations, took a lot of time to run. To solve this problem, we implemented the Gaussian Random projection on our data set and ran experiments to check the consistency of the results in the reduced dimension.

7 Dimensionality Reduction

We have a large dimension of 5968 feature in the database. We could apply Dimension Reduction algorithms to reduce the number of features. Various approaches like Principal Component Analysis, linear discriminant analysis, canonical correlation analysis, Discrete Cosine transform Method, Gauss Method, random Projection etc. are available for this purpose. We chose the random projection module as it implements a simple and computationally efficient way to reduce the dimensionality of data by trading a controlled amount of accuracy for faster processing times and smaller model sizes. This module construct two types of random matrix: Gaussian

random matrix and sparse random matrix. We use a Gaussian Random Projection to map the data from higher dimension to lower dimension.

Random Projection's main idea is based on the Johnson-Lindenstrauss theorem. Johnson-Lindenstrauss lemma states that a small set of points in a high-dimensional space can be embedded into a space of much lower dimension in such a way that distances between the points are nearly preserved. The Gaussian Random Projection uses this theorem and a value of epsilon as a measure of error allowed to obtain a matrix with random Gaussian distribution of features in the new dimension. Now the original feature matrix is mapped into this new feature matrix. Now since each iteration of this approach produces a slightly different random matrix, we report the observations of the classifiers based on averaged results over 10 iterations.

For reading the ARFF data in python, we used the liac-arff package. After importing the matrix, experiments were performed using the above operations and their results are reported in the table 6.

	TPR	ROC Area
Naive Bayes	61.54	0.625
J48	61.2	0.598
AdaBoost(Decision Stump)	51.29	0.606
AdaBoost(Naive Bayes)	61.18	0.616

Table 6

We can clearly see that the reduced dimension feature data set performed to a similar degree as compared to the original data set. The still poor values of TPR and ROC Area led us to believe that maybe more different features were needed to be added to our data set in order to improve the classification. Cosine similarity was one such feature which represents the similarity measure. It is explained in the next section and the results obtained on adding it to the data set.

```
#Code Extracts to read arff data
import arff # liac-arff package
dataset = arff.load(open('file.arff','r'))  # open the arff file
data = np.array(dataset['data']).astype(np.float32)  # obtain the data section
# of the arff file
data_vec = data[:,:-1] # obtain the feature vector except label
transformer = random_projection.GaussianRandomProjection(eps=0.25) # initialize Gaussian
# Random Projection
X = transformer.fit_transform(data_vec) # mapping to lower dimension
```

8 Cosine Similarity

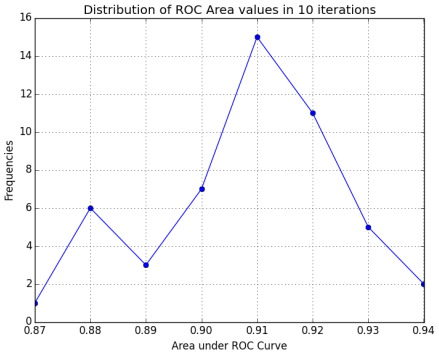
Another measure to quantify the similarity between two vectors is the Cosine similarity. Cosine similarity is the dot product of two vectors and measures the cosine of the angle between the two. This concept can be extended to text classification also. The dot product of the feature vectors of the PDF section and PPT section gives us the value of the cosine similarity of this pair and we add this value as a feature to our existing data set.

In order to implement cosine similarity, we need a common dictionary for the vectors of both PDF and PPT section. Hence we combine all the PDF text and PPT text in one TXT file. Using scikit-learn, a dictionary of all the text in the file was created. Now each section was converted into a feature vector representing tf-idf scores using the same dictionary. As a result, the feature vectors of both PPT and PDF sections were of the same size and built from the same key values. The cosine similarity was calculated, using the scikit-learn package on each pair of labelled PPT - PDF vector and then added as feature in our data set of that particular pair. Sub sampling and dimension reduction was also performed on the feature matrix before classification. Now testing of this new data set was done. The results are as reported in the table 7.

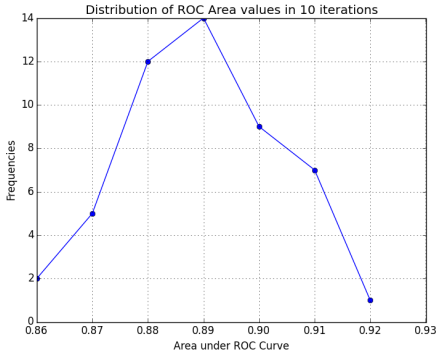
	TPR	ROC Area
Naive Bayes	79.88	0.908
AdaBoost(Decision Stump)	80.56	0.890

Table 7

The Area under ROC curve were generated over 10 iterations, each different due to Gaussian Random Projection using a 5-fold cross validation. The figure 1 shown below plot the various Area under ROC Curve values for the different classifiers above and the final value was reported as an average over all the points obtained.



(a) Naive Bayes



(b) AdaBoost(Decision Stump)

Figure 1: ROC Curve distribution over 10 iterations

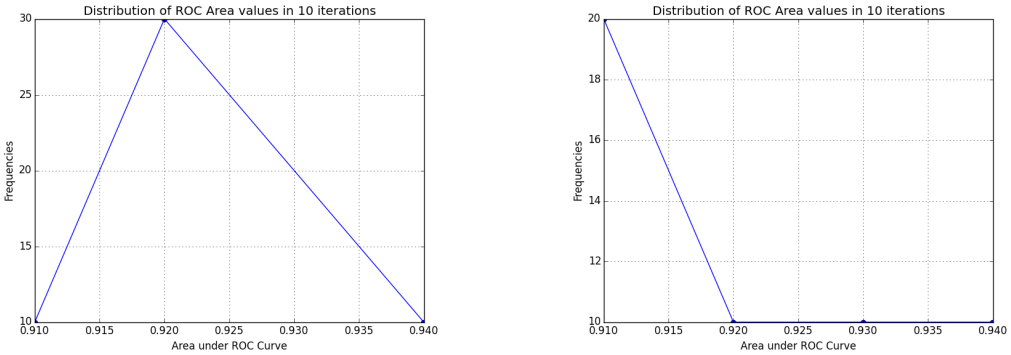
The experiments show a very high jump in Area under ROC Curve due to the addition of cosine similarity. Hence to investigate further we performed two more sets of experiments.

In the first experiment, we removed all the features except the cosine values and used the classifiers on only this 1 feature. Only the sub sampled data was used. The results are as reported in the table 8 and figure 2 represents the distribution of the Area under ROC Curve.

By now it was clear that the cosine similarity feature was the most important feature. So in our next experiment, we removed the cosine feature, performed the dimensionality reduction on the remaining feature set and then added the cosine

	TPR	ROC Area
Naive Bayes	78.54	0.922
AdaBoost(Decision Stump)	86.03	0.922
AdaBoost(Naive Bayes)	78.7	0.888
SMO	82.9	0.859

Table 8



(a) Naive Bayes

(b) AdaBoost(Decision Stump)

Figure 2: ROC Curve distribution over 10 iterations

similarity feature to the reduced dimension feature vector. Now we use classifiers on this new feature vectors. The results are reported in the table 9 and figure 3 shows the ROC Curves.

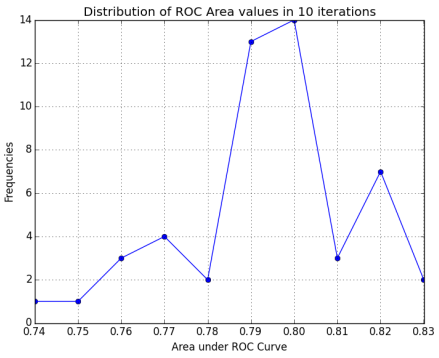
	TPR	ROC Area
Naive Bayes	75.42	0.794
AdaBoost(Decision Stump)	85.47	0.926
AdaBoost(Naive Bayes)	77.2	0.824
SMO	82.2	0.823

Table 9

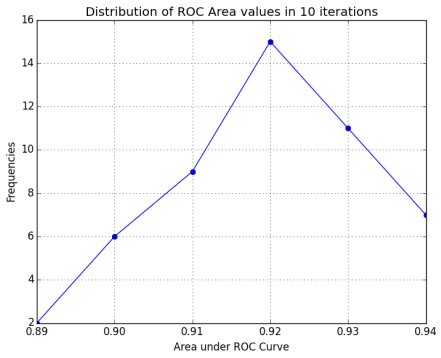
References

[1] B. Beamer and R. Girju. Investigating automatic alignment methods for slide generation from academic papers. *Proceedings of the Conference of the Association of Computational Linguistics*, 2009.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.



(a) Naive Bayes



(b) AdaBoost(Decision Stump)

Figure 3: ROC Curve distribution over 10 iterations

[3] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Pearson Prentice Hall, 2008.

[4] Hasida Koiti. Automatic slide presentation from semantically annotated documents. *Proceedings of the Workshop on Coreference and Its Applications*, 1999.

[5] T. Mikolov, M. Karafiat, L. Burget, J. Cernocky, and S. Khudanpur. Recurrent neural network based language model. *Proceedings of the Annual Conference of the International Speech Communication Association*, 2010.