A Semantic Triplet Based Story Classifier

by

Ravi Chandravadan Karad

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2013 by the
Graduate Supervisory Committee:

Hasan Davulcu, Chair
Steven Corman
Arunabha Sen

ARIZONA STATE UNIVERSITY

May 2013

ABSTRACT

Text classification, in the artificial intelligence domain, is an activity in which text documents are automatically classified into predefined categories using machine learning techniques. An example of this is classifying uncategorized news articles into different predefined categories such as "Business", "Politics", "Education", "Technology" , etc. In this thesis, supervised machine learning approach is followed, in which a module is first trained with pre-classified training data and then class of test data is predicted. Good feature extraction is an important step in the machine learning approach and hence the main component of this text classifier is semantic triplet based features in addition to traditional features like standard keyword based features and statistical features based on shallow-parsing (such as density of POS tags and named entities). Triplet {Subject, Verb, Object} in a sentence is defined as a relation between subject and object, the relation being the predicate (verb). Triplet extraction process, is a 5 step process which takes input corpus as a web text document(s), each consisting of one or many paragraphs, from RSS feeds to lists of extremist website. Input corpus feeds into the "Pronoun Resolution" step, which uses an heuristic approach to identify the noun phrases referenced by the pronouns. The next step "SRL Parser" is a shallow semantic parser and converts the incoming pronoun resolved paragraphs into annotated predicate argument format. The output of SRL parser is processed by "Triplet Extractor" algorithm which forms the triplet in the form {Subject, Verb, Object}. Generalization and reduction of triplet features is the next step. Reduced feature representation reduces computing time, yields better discriminatory behavior and handles curse of dimensionality phenomena. For training and testing, a ten- fold cross validation approach is followed. In each round SVM

classifier is trained with 90% of labeled (training) data and in the testing phase, classes of remaining 10% unlabeled (testing) data are predicted. Concluding, this paper proposes a model with semantic triplet based features for story classification. The effectiveness of the model is demonstrated against other traditional features used in the literature for text classification tasks.

DEDICATION

To my parents, family, friends and colleagues.

ACKNOWLEDGMENTS

I sincerely thank my advisor Dr. Hasan Davulcu for his continued guidance, support and encouragement during my masters and while writing this thesis. My sincere thanks to Dr. Steven Corman, to provide me an opportunity to work under his guidance, who have encouraged and guided me throughout the process. I also would like to thank Dr. Arunabha Sen for being on my thesis supervisory committee and for providing the guidance and the feedback on my work. Many thanks to Dr. Hessam Sarjoughian for accepting and responding quickly to the request, to attend thesis dissertation. I would like to thanks to all my lab-mates Betul, Ashok, Sedat, Ajay and all members of Centre for Strategic Communication (CSC) research lab of Hugh Downs School of Human Communication at Arizona State University for making this journey so exciting and making this a great learning experience. I am very grateful for the love and the unconditional support of my family, roommates and friends.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

LIST OF FIGURES

Chapter 1

INTRODUCTION

1.1 Statement of Purpose

Text classification is a process in which input text is classified into predefined categories. For example classifying news article into different categories such as “Business”, “Politics”, “Education”, “Technology” , etc. Text classification can be done manually as well as algorithmically. Algorithmic text classification, in the artificial intelligence domain, is an activity in which text documents are automatically classified into respective categories using machine learning techniques. There are two main subtypes of machine learning approach.

- Supervised Learning approach
- Unsupervised Learning approach

supervised learning approach involves training and testing phases. In training phase the module is trained with trained data which is correctly labeled standard sample data. In testing phase the class of unlabelled data is determined based on training data. Supervised learning approach has many applications like predicting whether certain customer is likely to purchase certain product, handwriting observation based on samples submitted by user and etc. Unsupervised learning approach does not consist training (annotated) data and it tries to find hidden pattern from test data exclusively. It involves algorithms like clustering and association rules. Example of unsupervised learning is search engine.

In this thesis, we are using supervised machine learning approach as we are predicting class of unlabelled data based on pre classified sample data. Good feature

extraction is an important step in machine learning approach and hence the main component of our text classifier is semantic triplet based features in addition of traditional features like standard keyword based features and statistical features based on shallow-parsing (such as density of POS tags and named entities). In this section we explain what are semantic triplets, extraction of triplets, formation of triplet based features, feature generalization, feature matrix formation, training & testing phases and SVM classifier.

We define a triplet in a sentence as a relation between subject and object, the relation being the predicate (verb). Extraction of Triplets is a process of finding significant information from an input text like subject (who), verb (doing what), direct object (to whom), Indirect Object (when and where). Triplet extraction, in addition removes irrelevant information such as stop words (a, an, the, he, she, etc) and irrelevant clauses. Hence with some modification, it can also be used in applications like summarization of news articles and question-answering tool. We use Semantic Role Labeler (SRL), a shallow semantic parser to extract sets of the form {subject, predicate, object} out of syntactically parsed sentences. If we input our triplet extractor a very simple paragraph from news article, “Musharraf began demolishing homes, making arrests, and killing innocent people”, then extracted triplets are: {“Musharraf ” “demolish ” “homes ”} {“Musharraf ” “make” “arrests ”} {“Musharraf ” “kill” “innocent people ”}.

"Triplet based feature formation" is the next step in which, for each verb (V) mentioned in the above triplets, we stem and aggregate its arguments corresponding to its SUBJECTs, OBJECTs and PREPOSITIONs to generate following set-valued “semantic verb features” by using the training data:

f_k : $Label_i$. Verb . Subject = { Argument list }

f_{k+1}: $Label_i$. Verb . Object = { Argument list }

f_{k+2}: $Label_i$. Verb . Preposition = { Argument list }

Where, f_k is a feature ID, $Label_i$ is i^{th} label of the paragraph of the input training corpus , and “Argument list” refers to list of relevant subject, object or prepositional arguments of a verb.

For example, for the triplet {“Musharraf ” “demolish ” “homes ”}, say coming from a paragraph having category “Label1”, triplet features are:

f_1 :$Label_1$. demolish. Subject = { Musharraf }

f_2: $Label_1$. demolish . Object = { homes }

Triplet features in the testing phase, do not have label associated with them because label is the entity that we have to calculate for testing data. So, in case of test data features “$Label_i$” in the above feature keys is replaced with paragraph id (PID).

The next step "triplet feature generalization" is an important step in classification process where number of features are reduced. Reduced feature representations not only reduce computing time but they may also yield to better discriminatory behavior. Owing to the generic nature of the curse of dimensionality it has to be assumed that feature reduction techniques are likely to improve classification algorithm.

Next step, which is "feature matrix formation", is a process in which two feature matrices corresponding to training and testing phase, are formed. Training feature matrix is 2 dimensional matrix with rows as paragraphs ids and columns as feature ids. Each row in matrix consist of paragraph id (PID) as the first column and triplet features normalized as 1 or 0s. Testing feature matrix is same as training feature matrix with an exception that

training feature matrix has last column marked as "Label" and testing feature matrix does not have "Label" column.

In training and testing step, we follow 10 fold cross validation. In each iteration we train the SVM classifier with all categories of 90% of labeled data and in testing phase, we test remaining 10% of unlabeled data.

1.2 Scope

The scope of the thesis is to classify web text documents into two categories named “Story” and “Non-story”. In chapter 2, we elaborate significance, definition and background of these two categories. In this thesis, we utilize a corpus of 16,930 paragraphs where 3, 301 paragraphs coded as stories, and 13,629 paragraphs coded as non-stories by domain experts to develop a story classifier. Training data is a collection of Islamist extremist texts, speeches, video transcripts, forum posts, etc., collected in open source. We investigate the utility of standard keyword based features, statistical features that can be extracted using shallow-parsing (such as density of POS tags and density of named entities), and a new set of semantic features in development of a story classifier.

1.3 Motivation

Our study is motivated by the observation [3] that interrelated stories that work together as a system are fundamental building blocks of (meta-) narrative analysis. We focus on discriminating between stories, and non-stories. The main purpose of developing an automated story classifier is to reduce the human dependency to annotate story and non-stories.

The main contribution of this thesis is the introduction of a new set of semantic features based on related linguistic subject, verb, object categories that we named as *triplet based verb features* which are motivated by the definition of "story" as "actors taking actions that culminate in resolutions". Our proposed semantic features are based on suitable aggregation and generalization of <Subject, Verb, Object> triplets that can be extracted using a shallow-parser. Experimental results (see Table 5) show that a combination of statistical part-of-speech (POS) and named-entity (NE) features, with semantic triplet-based features achieves highest accuracy with a Support Vector Machine (SVM) based classifier. We obtained precision of 73%, recall of 56% and F-measure of 0.63 for minority class (i.e. stories) which indicates a 161% boost in recall, and an overall 90% boost in F-measure with negligible reduction in precision through the utility of triplet based features over standard keyword based features.

1.4 Outline

The rest of the report is organized as follows. Chapter 2 gives background literature. Chapter 3 mentions related work. Chapter 4 describes system architecture. Chapter 5 elaborates semantic triplet extraction process. Chapter 6 explains generalized verb feature and feature reduction. Chapter7 describes experimental evaluation. Chapter 8 concludes the thesis with conclusion and future work

Chapter 2

BACKGROUND LITERATURE

A story is defined as “an actor(s) taking action(s) that culminates in a resolution(s).” Personal narratives are powerful sources of persuasion, none more so than stories that cultural heroes tell about their own lives [1]. Whether their account retells the story of a great athlete or actor or celebrity or terrorist, fans are drawn to these accounts as moths to bright lights. In part this is because the stories themselves can be quite interesting, and in part because readers often closely want to in some way identify their own lives with the life stories of their heroes [2]. An investigation of terrorist narrative communication through an in-depth examination of extremists published autobiographies and interviews can be helpful in understanding mindsets and motivation behind terrorist activities. In addition, the analysis of terrorist narratives across geographical regions holds the potential to illustrate cultural differences, as well as to illustrate how telling their own stories serves to recruit and assimilate outsiders into local political groups and extremist organizations. But the problem with analysis of extremist text is that it needs many human annotators to extract stories and non stories from different sources. The main purpose of developing an automated story classifier is to reduce the human dependency to annotate story and non-stories.

A story is comprised of three components. First, there must be an actor or actors telling the story implicitly or explicitly. This can include politicians, mujahedeen, everyday people and so on. Second, the actors must be performing actions. This can include fighting, preparing for a battle, talking to others and so on. Third, the actors actions must result in a resolution. Resolutions can include a new state of affairs, a new

equilibrium created, a previous equilibrium restored, victory and so on. Besides, stories usually have story worlds, or worlds where the stories are taking place. Story worlds are not fictional universes, but rather environments in which the story takes place.

Story example:

"They have planted your remains in the sands like a flag. To motivate the people morning and night, woe unto them, they have raised a beacon of blood To inspire tomorrow's generation with hate and dislike".

A non-story paragraph is one, among the categories Exposition, Supplication, Question, Annotation, Imperative, Verse or Other. Below paragraph is coded as "Non-Story" because there is no explicit resolution. There are only hypothetical resolutions.

Non-Story example:

"Let the soldiers of this Administration go to hell. Petraeus and Bush are trying to convince the Americans that their salvation will begin six weeks from next July. In fact even if Bush keeps all his forces in Iraq until doomsday and until they go to hell, they will face only defeat and incur loss, God willing."

Chapter 3

RELATED WORK

Computational models of stories have been studied for many different purposes. R.E. Hoffman et al. (2011) [4] modeled stories using an artificial neural network. After the learning stage, they compare the story- recall performance of the neural network with that of schizophrenic patients as well as normal controls in order to derive a computational model which matches the illness mechanism. The most common form of classification applied for stories tackles the problem of mapping a set of stories to predefined categories. One of the popular applications is the classification of news stories to their topics [5], [6].

Gordon investigated the problem of detecting stories in conversational speech [7] and weblogs [8] and [9]. In [7], the authors train a Naive Bayes classifier to categorize the transcribed text of a speech into story and non-story categories. Using word-level unigram and bigram frequency counts as feature vectors, they reported results for the classification of a speech as a story with 53.0% precision, 62.9% recall and 0.575 F-measure. For weblogs, in [8], they incorporated techniques for automatically detecting sentence boundaries to their previously used text features to train a Support Vector Machine classifier. After smoothing the confidence values with a Gaussian function, they achieved 46.4% precision, 60.6% recall and 0.509 F-measure.

In Gordon and Swanson's most recent work on story classification [9], they used a confidence-weighted linear classifier with a variety of lexical features, and obtained the best performance with unigrams. They applied this classifier to classify weblog posts in

the ICWSM 2009 Spinn3r Dataset, and they obtained 66% precision, 48% recall, and F-measure of 0.55.

Chapter 4

SYSTEM ARCHITECTURE

Figure 1. System Architecture

4.1 Data Collection

Our corpus is comprised of 16,930 paragraphs from extremist texts collected in open source. Stories were drawn from a database of Islamist extremist texts. Texts were selected by subject matter experts who consulted open source materials, including www.opensource.gov, private collection/dissemination groups, and known Islamist extremist web sites and forums. Texts come from groups including al-Qaeda, its affiliates, and groups known to sympathize with its cause and methods. The subject matter experts selected texts which they believe contained or were likely to contain stories, defined as a sequence of related events, leading to a resolution or projected resolution.

Extremists' texts are rarely, if ever, composed of 100% stories, and indeed the purpose of this project is to enable the detection of portions of texts that are stories.

Accordingly, we developed a coding system consisting of eight mutually-exclusive and exhaustive categories: story, exposition, imperative, question, supplication, verse, annotation, and other along with definitions and examples on which coders could be trained. After training, coders achieved reliability of Cohen's Kappa = 0.824 (average across eleven randomly sampled texts). Once reliability of the coders and process was established, single coders coded the remainder of the texts, with spot-check double coding to ensure reliability was maintained.

The Cohen's Kappa measure represents how two observers agree on sorting items into different categories. The observers can be human or machine. The range of Cohen's Kappa varies between 0 and 1. Fleiss [10] characterizes Kappa range over 0.75 as excellent, range between 0.40 to 0.75 as fair to good, and less than 0.40 as poor. Hence coders' reliability of 0.824 falls into the range of excellent. After training and testing with ten-fold cross-validation, we calculated the agreement between classifier algorithm and the human coders as Cohen's Kappa = 0.48, which falls into the range of fair to good.

4.2. Human Annotation: Story vs. Non-Story Coding

Stories are differentiated from non-stories as follows:

- Stories will have a lower proportion of stative verbs than non-stories because they describe actions.
- Stories will include more named entities, especially person names, than non-stories. Stories will use more personal pronouns than non-stories.
- Stories may include more past tense verbs (i:e:, X resulted in Y, X succeeded in doing Y, etc.) than non-stories.

- Stories may repeat similar nouns. For example, “mujahedeen” may be mentioned in the beginning of the story and then again at the end of the story.
- Paragraphs with stories in them have different sentence lengths than paragraphs without stories in them.

Human annotators were given training before they are assigned the task to manual classification of story vs. non-story. The training procedure is explained as below:

1. All trainees read the training manual which elaborate rules about story and non-story classes with examples (This is explained in chapter 2, “Background Literature”).

2. Once the training manual has been read, trainees meet with the trainers to discuss any questions they may have in regard to the training manual. Trainers answer the questions by referencing past experience, already coded text, and/or the training manual. Trainers are individuals that are familiar with the story coding procedures and who have shown reliable coding skills (70% reliability or greater) over time.

3. Separately, each trainee codes 5 texts that have already been coded by the trainers. All of the trainees code the same 5 texts. Trainers assign texts to be coded by using the online system. (Intranet website)

4. After the trainees have finished coding their texts, the trainers use the online system to assess the reliability scores of each of the trainees.

5. Trainers present the results of the coded texts to the trainees. Trainers go over every error that was made, explain what the correct codes should have been, and

provide reasons as to why a particular code should have been chosen over another. Additionally, trainers answer any questions the trainees may have.

6. Trainees then, separately, code 10 other texts that have already been coded by the trainers. All of the trainees code the exact same 10 texts, and the 10 texts are different than any text already coded by the trainees. Trainers assign texts to be coded by using the online system (Intranet website).

7. After the trainees have finished coding their texts, the trainers use the online system to assess the accuracy and reliability scores of each of the trainees.

8. Trainees that have received reliability scores of 70% or greater have passed the training course and are permitted to code new texts. Trainees that have received reliability scores below 70% are required to repeat steps 5-7 until they receive reliability scores of 70% or greater.

4.3. Data Preprocessing

4.3.1. Named Entity Recognition Tagger

Named entity recognition (NER) [11] (also known as entity identification or entity extraction) is a subtask of information extraction that seeks to locate and classify atomic elements in text into predefined categories such as persons, organizations, locations. Research indicates that even state-of-the-art NER systems are brittle, meaning that NER systems developed for one domain do not typically perform well on other domains [12].

For the purpose of annotating the entities found within the texts belonging to extremist narratives, we used the most popular publicly available NER libraries. We

evaluated three libraries: Stanford Named Entity Recognizer [13], Illinois Named Entity Tagger [14] and an online web service provided by Open Calais [15].

A set of six documents belonging to extremist narratives were manually annotated by a specialist as a person, a location or an organization. Next, the same set of documents was annotated using NER libraries in order to determine their accuracy. F-measure was used to measure their accuracy as shown in Table 1.

Text-ID	**DNERT**	**Stanford**	**Illinois**	**Open Calais**
1	0.592	0.355	0.463	0.312
2	0.567	0.587	0.549	0.164
3	0.652	0.627	0.574	0.247
4	0.837	0.867	0.867	0.591
5	0.720	0.686	0.483	0.459
6	0.505	0.446	0.651	0.416
Average	0.644	0.594	0.597	0.364

Table 1. NER Tagger F-measures

We also developed a consensus analysis based algorithm, which we named as ‘Democratic NER tagger’, using the output of all three taggers as follows:

1. For a particular text document to be annotated for named-entity tags, invoke each NER tagger (Stanford, Illinois and Open Calais).

2. For an annotated entity/word determine the category (Person, Organization, Location) assigned by each NER tagger.

3. If the phrase is classified by only one tagger, then assign it as the final tag.

4. If all taggers disagree on the category of a phrase, then we pick the final category according to the accuracy of the taggers as follows: Illinois NER has the highest accuracy for Locations and Organizations; Stanford NER has the highest accuracy for Persons.

5. If two out of three NER taggers agree on the predicted category of a phrase, then the final category is determined by majority agreement.

Within the six documents, specialist annotated 308 organization names, 259 location names, and 127 person names. Table 1 summarizes the accuracies of the software libraries, as well as the accuracy of our simple democratic NER tagger (DNERT) which relies on all of them. Overall, our democratic NERT achieves the highest performance compared to individual NER taggers.

4.3.2 Named Entity Standardization

The extremist texts under consideration are collected from various social media sources (i.e. blogs, organizations' websites and RSS feeds). Since these are all human generated documents, they are rife with misspellings and aliases of named entities. For example, the person entity 'Osama Bin Laden' is sometimes referred to as 'Bin Laden', 'Sheikh Osama', or it is sometimes misspelled as 'Osamma Bin Laden' or 'Usama'. In order to standardize the usage of the entities and improve the accuracy of classifier dependent on the entity features we have came up with a two step named entity standardization process:

4.3.2.1 Misspelling Correction Step

The named entity spelling is corrected using a lookup with the Google's 'Did u mean?' feature. This process enables us to identify the most correct spelling of a named entity. E.g. An entity named 'Osamma bin laden' is corrected as 'Osama bin laden'.

4.3.2.2 Standardization for Aliases

We query the RDF data stores of DBpedia [16] in order to find a standardized name for all location, organization and person entities. DBpedia data set has a public SPARQL endpoint available at: http://DBpedia.org/sparql. The DBpedia OWL ontology for named entity types have following properties where known aliases are stored:

dbpprop:alternativeNames, dbpprop:name, foaf:name.

By querying the alternative names we are able to obtain the standard name for each entity.

Occurrences	**Person**	**Organization**	**Location**
Total	5015	2456	4279
Distinct	332	200	290
Standardized	72	26	30
Accuracy	65 (90.3%)	24 (92.3%)	28 (93.3%)

Table 2. Named-Entity Correction and Standardization Results

Table 2 summarizes the results for the named entity correction and standardization algorithm. First row shows the total number of occurrences of each type of named entity in our document corpus. Second row shows the counts of distinct entities

by their type. Third row shows the number of distinct named entity corrections made through above spelling correction and alias standardization procedure. The changes were manually evaluated by a human annotator to verify their accuracy. As seen in the final row of the table, out of 72 person name changes, 65 were accurately standardized by the algorithm, providing 90% accuracy for person entities. Similarly, for organization entities the correction and standardization accuracy is around 92%, and for location entities it is around 93%.

4.4. Feature Extraction

Above observations made in chapter 3 motivates following standard keyword based features, statistical features based on shallow-parsing, and a new set of semantic features for the development of a story classifier:

4.4.1 Keywords

TF/IDF measure [17] is calculated for each word contained in the whole paragraph set. Then a certain number of terms, in our case 20, 000, with the top TF/IDF values are selected as features. Then term-document frequency matrix is created out these keyword features.

4.4.2 Density of POS Tags

Part of Speech (POS) Tag Ratios [18] for each document is calculated with respect to numbers of tokens.

4.4.3 Density of Named Entities

Named Entity (NE) Tag frequency [13] per document is calculated. The tags are Person, Location and Organization, explained in section 4.3 above.

4.4.4 Density of Stative Verbs

Some other statistical features are also included in all experiments, such as the number of valid tokens and the ratio between observed stative verbs and total number of verbs in a paragraph.

4.4.5 Semantic Triplets Extraction

We present our semantic triplet extraction method in chapter 5. Semantic triplets are the news features we discovered and are building block of our classifier. We also discuss how triplets from stories and non-stories are aggregated and generalized to form memory-based features for verbs.

4.5 Support Vector Machine (SVM) Classifier

SVM [19] is a supervised machine learning classification algorithm which makes use of a hyperplane to separate the data into two categories. There are other classifier algorithms available like k-means, Naive Bayes, PageRank, AdaBoost, kNN and others. Because it offers most robust and accurate method among all well known classification algorithms, we chose support vector machine. SVM [31] has many advantages over other classifiers including, always having a global optimum and usually needs only a few parameters (named kernel parameters) to be set for good performance. This simplicity does not come with performance degradations. In fact, many studies [31] showed that linear SVMs perform very well for text classification tasks. Support vector machines algorithms are also referred as max margin algorithms because SVMs looks for linear boundaries that best separate given classes from each other.

Let us be given a set of observations, O, where each observation is a pair (x, y) where $x \in R_n$ is a vector in a given n-dimensional vector space and y is the label associated with the observation. For example, assume, in training phase we considered m text paragraphs and extracted total n features, then observation O is a set of these m text paragraphs, a feature vector x for a text paragraph is a vector formed by n features and the label y is the label of that paragraph tagged by human annotator, which is either story or non-story.

The goal of the SVM is to partition the space using these observations (i.e., the training data) into regions, such that each region contains observations with a single label as shown in Figure 2.

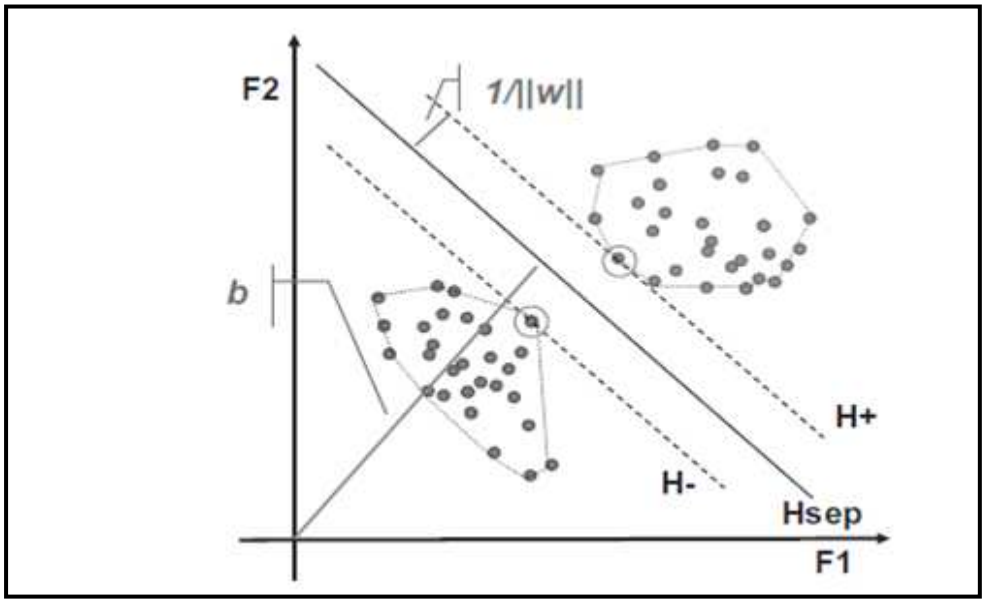

Figure 2: Linear SVM classifier in 2-D feature space [31]

Assume, each feature vector x, corresponding to a paragraph has one of two values “1” for story and “-1” for non-story. Also assume that there is at least one plane in n dimensional space which separate these two sets of feature vectors from each other.

This plane is called linearly separable plane and set of these features vectors is called linearly separable. If there is one plane separating these feature vectors, it is possible that there could be other planes which linearly separate these feature vectors. Because there are many such linear hyper planes SVM additionally finds the best hyperplane by maximizing the margin between the two classes (story and non-story). The margin is the amount of space, or separation between the two classes, defined by the hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points of two feature vectors' sets, as seen in the figure 2.

SVM is originally proposed as a linear classifier [20] but later improved by the use of kernel functions to detect nonlinear patterns underlying the data [21].There are various types of kernel functions available [22]. In this study, we use RBF kernel defined as

$$K(x_i, x_j) = e^{\|xi-xj\|} \qquad \text{where } x_{i,j} \text{ are data points[23].}$$

RBF kernel is non linear separable kernel but it adds curve around the data points (feature vectors) as shown in figure 3.

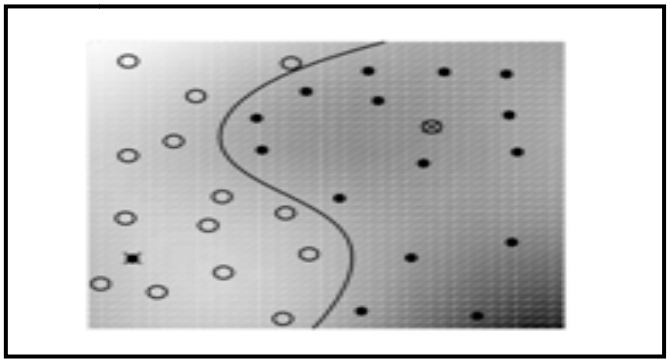

Figure 3: RBF Kernel

There are various software package available for SVM. We used LIBSVM [32] tool with Matlab version. Below Matlab code is run to get the classified data.

```
[ nrow ncol ] = size( data );
ordering = randperm( nrow );
data = data( ordering, : );
tr_data = data(  1 : ceil( nrow * 0.9 ), 2 : ncol - 1 );
labels = data( :, ncol );
[ tr_nrow tr_ncol ] = size( tr_data );
ts_data = data( tr_nrow + 1 : end, 2 : ncol - 1 );
tr_labels = labels(  1 : tr_nrow );
ts_labels = labels(   tr_nrow + 1 : end );
tr_data_sp = sparse( tr_data );
ts_data_sp = sparse( ts_data );
model = svmtrain( tr_labels, tr_data_sp );
[predict_label, accuracy, dec_values] = svmpredict( ts_labels, ts_data_sp,
model  );
```

Multiclass SVM

As explained in section 1.2 our scope is limited to classify data into two classes named story and non-story. But if the input data has to be classified in more than 2 categories then we can use multiclass SVM approach. Below are multiclass approaches we can choose from.

- **One-against-all classification**, in which there is one binary SVM for each class to separate members of that class from members of other classes.
- **Multi class ranking SVMs**, in which one SVM decision function attempts to classify all classes.
- **Pairwise classification**, in which there is one binary SVM for each pair of classes to separate members of one class from members of the other.

4.6. Training and Testing

Our text classifier uses supervised machine learning approach which involves training and testing phases. In training phase the classifier is trained with pre classified sample data. In this phase input text paragraphs are processed to extract features and build a feature matrix. SVM classifier is trained by inputting this feature matrix. In testing phase, we take test paragraph, extracts it's features and build test feature matrix and input to SVM classifier. SVM classifier based on training data assigns the label to test paragraph.

The corpus contains 1,256 documents, each containing both story and non-story paragraphs. There are a total of 16,930 paragraphs, where 13,629 paragraphs classified reliably as non-stories, and 3,301 paragraphs classified as stories by domain experts.

Cross-validation [33]

It is a technique in which documents to classify are randomly divided into m subsets of equal size. One iteration of cross validation involves processing of one subset, also called as training subset, and validating the analysis of other subset, also called as

testing subset. Multiple iterations of cross-validation are performed using different partitions to reduce variability, and the validation results are averaged over the rounds. It is mainly used in settings where the goal is prediction, and one wants to estimate how accurately a predictive model will perform in practice. There are 4 common types of cross validation techniques explained as below:

1. k-fold cross validation

In this type of cross validation, the input documents are divided into k different subsets. One iteration consist of training data as k-1 subsets and testing data includes remaining one subset. There are k such iterations performed and final result of cross validation is calculated by averaging results of these k iterations. Value of k is flexible but normally k=10 is selected and called as Ten-fold cross validation.

2. 2-fold cross validation

It is simplest version of k-cross validation method where k=2. In first iteration, input data is divided into two subsets. The first subset is trained and second subset is tested. In second iteration, second subset is trained and first subset is tested.

3. Repeated random sub-sampling validation

This method randomly divide the dataset into training and testing data. For each such division, the model is fit to the training data, and predictive accuracy is assessed using the testing data. The results are then averaged over the division. The advantage of this method (over *k*-fold cross validation) is that the proportion of the training/validation split is not dependent on the number of iterations (folds). The disadvantage of this method is that some observations may never be selected in the validation subsample,

whereas others may be selected more than once. In other words, validation subsets may overlap.

4. Leave-one-out cross-validation

Leave-one-out cross-validation involves using a single observation from the original sample as the validation data, and the remaining observations as the training data. This is repeated such that each observation in the sample is used once as the validation data. This is the same as a *K*-fold cross-validation with *K* being equal to the number of observations in the original sampling.

In our evaluations, we performed 10 fold cross validation (a type of k-fold cross validations) with the document files as follows; we break documents into 10 sets of size n/10, where n is total number of documents (1,256). During the training phase, both story and non-story paragraphs from 9/10 documents are used as the training set, their features are extracted, and a classifier is trained. During the testing phase, the remaining 1/10th of the documents are used; the features for both stories and non-stories are extracted, and matched to the features extracted during the training phase. Doing this evaluation, we are ensuring that training and test data features are in fact coming from different documents. We calculate precision, recall for each iteration of the 10 fold cross validation and we report mean precision, recall for both stories and non-stories.

Chapter 5
SEMANTIC TRIPLET EXTRACTION

Semantic triplet extraction is a feature extraction technique based on semantic triplets mentioned in system architecture (figure 1). Triplet based feature is the main building block which boosts precision and recall of our classifier. This chapter explains the pipeline of triplet feature extraction process.

Triplet extraction pipeline takes input as a web text document(s) each consisting one or many paragraphs. Pronoun Resolution step uses heuristic approach to identify the noun phrases referenced by the pronouns in the paragraph and replaces those pronouns with relevant nouns. The pronoun resolved output is fed to next step “SRL Parser”. SRL Parser is shallow semantic parser and converts the input paragraph into annotated predicate argument structure. The output of SRL parser is processed by Triplet Extractor algorithm which forms the triplet in the form <Subject, Verb, Object>. These triplets are persisted in database so we can reuse it later instead of reinventing the wheel. The output of Triplet Extractor is inputted to Propbank tagging step where predicates (verbs) arguments are tagged with their semantic roles using propbank database and also sense ID is assigned to a predicate. Propbank tagging step is very important to generalize and reduce number of feature to avoid curse of dimensionality phenomena.

We follow a standard verb-based approach to extract the simple clauses within a sentence. A sentence is identified to be complex if it contains more than one verb. A simple sentence is identified to be one with a subject, a verb, with objects and their modifying phrases. A complex sentence involves many verbs. We define a triplet in a sentence as a relationship between a verb, its subject and object(s). Extraction of triplets

[24], [25] is the process of finding who (subject), is doing what (verb) with/to whom (direct objects), when and where (indirect objects/and prepositions).

Triplet extraction utilizes the information extraction pipeline shown in Figure (4).

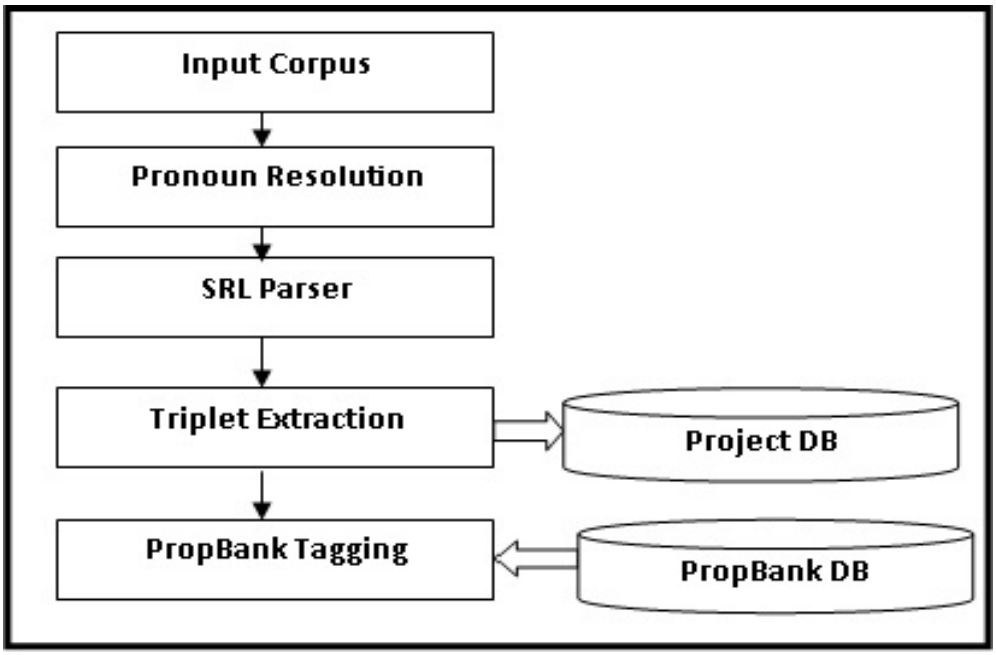

Figure 4. Triplet Extraction Pipeline

5.1 Input Corpus

Input corpus contains 1,256 documents, each containing both story and non-story paragraphs. There are a total of 16,930 paragraphs, where 13,629 paragraphs classified reliably as non-stories, and 3,301 paragraphs classified as stories by domain experts.

Our corpus, comprised of 16,930 paragraphs, is mainly a collection of Islamist extremist texts, speeches, video transcripts, forum posts, etc., collected from open source. Stories were drawn from a database of Islamist extremist texts. Texts were selected by subject matter experts who consulted open source materials, including www.opensource.gov, private collection/dissemination groups, and known Islamist

extremist web sites and forums. Texts also come from groups including al-Qaeda, its affiliates, and groups known to sympathize with its cause and methods. The subject matter experts selected texts which they believe contained or were likely to contain stories, defined as a sequence of related events, leading to a resolution or projected resolution.

5.2 Pronoun Resolution

Interactions are often specified through pronominal references to entities in the discourse, or through coreferences where, a number of phrases are used to refer to the same entity. Hence, a complete approach to extract information from text should also take into account the resolution of these references.

Our pronoun resolution module [26], [27] uses a heuristic approach to identify the noun phrases referred by the pronouns in a sentence. The heuristic is based on the number of the pronoun (singular or plural) and the proximity of the noun phrase. The closest earlier mentioned noun phrase that matches the number of the pronoun is considered as the referred phrase.

The input corpus is first preprocessed with pronoun resolution before feeding it to SRL. The advantage of adding this component improves the performance, explained as below. In the triplet extraction process, extractor removes stop word from the input paragraphs. All the pronouns are considered as stop words and are removed by stop word scan. Hence it is needed to replace the pronouns with referenced entities, before inputting to triplet extractor. The guiding principle behind pronoun resolution tool is that a pronoun appears closest to the proper noun to which it refers, here closest refers strictly to word

distance. Most often, a pronoun is used immediately, following the proper noun to which it refers for both the sake of simplicity as well as ease of understanding. However, this is not always the cause and will sometimes lead to a pronoun being resolved to the incorrect entity. Hence, selecting a good pronoun resolution algorithm is very important.

For our triplet extractor, we analyzed different Pronoun resolution and coreference resolution tools like JavaRap, Gate, Illinois Coreference Resolution, Stanford co-reference resolution and conclude that Stanford is the best one. Stanford coreference resolution has F-measure around 80% [26], [27] . But the problem with Stanford coreference resolution tool is that it does not have in-built pronoun resolution feature. So we have built our own pronoun resolution on the top of Stanford coreference resolution tool, by modifying its source code.

Example:

Input to Pronoun Resolution :

"America commissioned Musharraf with the task of taking revenge on the border tribes, especially the valiant and lofty Pashtun tribes, in order to contain this popular support for jihad against its crusader campaign. So he began demolishing homes, making arrests, and killing innocent people. He, however, pretends to forget that these tribes, which have defended Islam throughout its history, will not bow to US".

Output of Pronoun Resolution :

"America commissioned Musharraf with the task of taking revenge on the border tribes, especially the valiant and lofty Pashtun tribes, in order to contain this popular support for jihad against its crusader campaign. So Musharraf began demolishing homes,

making arrests, and killing innocent people. Musharraf, however, pretends to forget that these tribes, which have defended Islam throughout its history, will not bow to US".

5.3 Semantic Role Labeler (SRL) Parser

SRL parser [28] is the key component of our triplet extractor. To extract the subject-predicate-object from an input sentence, important step is identifying these elements in a sentence and parse it. SRL parser does exactly the same. SRL is proprietary software developed by Illinois research group and its shallow semantic parser. The goal of the semantic role labeling task is to discover the predicate-argument structure of each predicate that fill a semantic role and to determine their role (Agent, Patient, Instrument etc). As shown in the following example, SRL is robust in identifying verbs, their arguments and argument types accurately in the presence of syntactic variations.

For example, given a sentence *He gave his laptop to his friend in Chicago* the goal is to identify different arguments of the verb predicate left and produce the output:

[A0 $_{\text{He}}$] [V $_{\text{gave}}$] [A1 $_{\text{his laptop}}$] [A2 $_{\text{to his friend}}$] [AM-LOC $_{\text{in Chicago}}$].

Here A0 represents the giver (subject) , A1 represents the thing given (direct object), A2 represents the recipient (indirect object), AM-LOC is an adjunct indicating the location of the action (preposition) , and V determines the boundaries of the predicate which is important when a predicate contains many words, e.g., a phrasal verb.

5.3.1 SRL Annotations

Numbered arguments (A0-A5, AA):

Arguments define verb-specific roles. They depend on the verb in a sentence. There are total 6 different types of arguments A0, A1, A2, A3, A4, A5 and AA. The most

frequent roles are A0 and A1 and, commonly, A0 stands for the agent and A1 corresponds

to the patient or theme of the proposition. The distribution of these arguments is unbalanced and out of total arguments, A0-A5, AA occupies 71.26% of total arguments (A0 =25.39 %, A1 =35.19%).[28]

Adjuncts (AM-):

General arguments that any verb may take optionally. There are 13 types of adjuncts:

AM-ADV - general-purpose,

AM-MOD - modal verb,

AM-CAU - cause,

AM-NEG - negation marker,

AMDIR - direction,

AM-PNC - purpose,

AM-DIS - discourse marker,

AM-PRD - predication,

AM-EXT - extent,

AM-REC - reciprocal,

AM-LOC - location,

AM-TMP - temporal,

AM-MNR - manner.

References (R-)

Arguments representing R-arguments realized in other parts of the sentence. The label is an R- tag prefixed to the label of the referent, e.g. [A1 The Laptop] [**R-A1** which] [A0 he] [V gave] [A2 to his friend] was old.

5.3.2 SRL System Architecture [28] **:**

SRL works in four stages, starting with pruning of irrelevant arguments, identifying relevant arguments, classifying arguments as A0, A1, A2, AM-LOC and inference of global meaning.

Pruning -

This is the first stage in the SRL architecture. Pruning stage is used to filter out simple constituents that are very unlikely to be arguments using heuristic. It uses recursive approach which starts with a verb and returns all siblings of that verb. Then it goes to parent of these verbs. Next, for this parent root, it finds all of its siblings and continues this process until it reaches root verb.

Argument Identification -

This stage utilizes binary classification to identify whether a candidate is an argument or not. The classifiers are applied on the output from the pruning stage. A simple heuristic is employed to filter out some candidates that are obviously not arguments. Candidates are considered valid arguments if they do not violate following constraints.

- Argument cannot overlap with the predicate.

- If a predicate is outside a clause, its arguments cannot be embedded in that clause.
- Arguments cannot exclusively overlap with the clauses.

When the argument boundaries are known, the performance of the full parsing based SRL system is about the same as the shallow parsing based SRL system.

Argument Classification -

This stage assigns labels to the argument candidates identified in the previous stage. It uses multi-class classifier trained to explore the type of the argument. In addition, it marks arguments as null, if that argument is not considered as a valid argument in the previous stage. The learning algorithm used for argument classification and identification is based on little variation of “Winnow update rule”, implemented in SNoW, a multi-class classifier which is tailored for large scale learning tasks.

Inference -

In the previous stages, decisions were always made for each argument independently, ignoring the global information across arguments in the final output. The purpose of the inference stage is to incorporate such information, including both linguistic and structural knowledge. This knowledge is useful to resolve any inconsistencies of argument classification in order to generate final legitimate predictions.

5.3.3 Performance Comparison of SRL parser against other parsers

This section describes how SRL achieves high performance compared with other parsers. Inclusion of this section is important as this is the key component which helps

our triplet extractor to achieve high performance compared with other triplet extractor (based on other parsers like LinkParser or OpenNLP). The SRL is achieving high performance because of introduction of " Inference " component in its architecture. The importance of global inference to good performance is characterized by rich structural and linguistic constraints among the predicted labels of the arguments. The linear programming based inference procedure is a powerful and flexible optimization strategy that finds the best solution subject to these constraints. This approach is used to resolve conflicting argument predictions in an individual system and also serve as an effective and simple approach, resulting in a significant improvement in performance. Following figure (5) shows the precision and recall comparisons between different parsers.

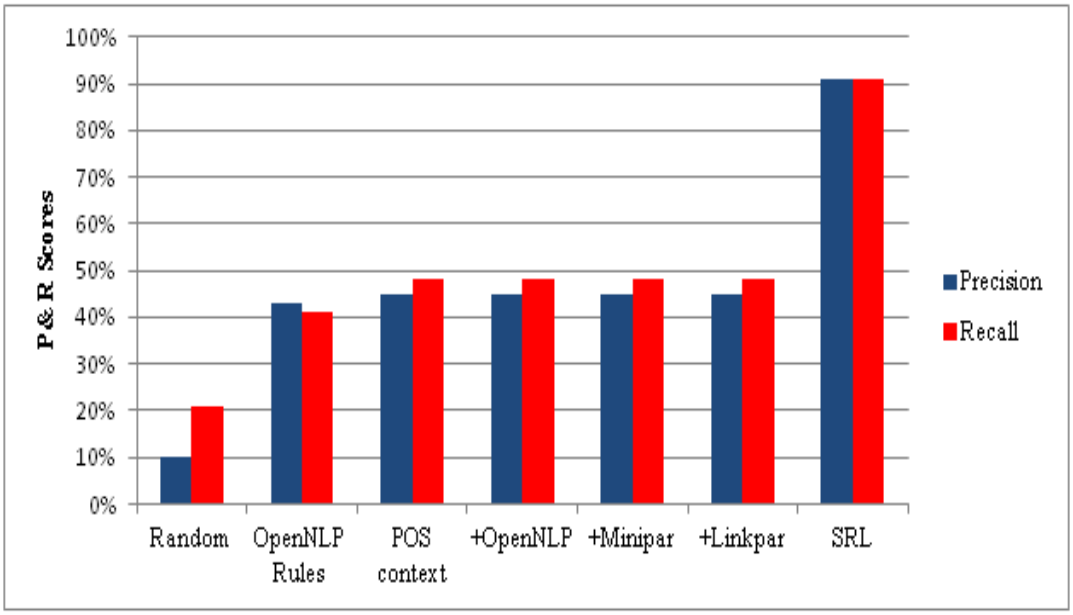

Figure 5: Comparison between different parsers [34]

Example:

Input to SRL parser:

"Therefore, America commissioned Musharraf with the task of taking revenge on the border tribes, especially the valiant and lofty Pashtun tribes, in order to contain this popular support for jihad against its crusader campaign. So Musharraf began demolishing homes, making arrests, and killing innocent people. Musharraf, however, pretends to forget that these tribes, which have defended Islam throughout its history, will not bow to US".

Output of SRL parser:

```
(S1 (S (ADVP (RB Therefore))
       (, ,)
       (NP (NNP America))
       (VP (VBD commissioned)
           (NP (NNP Musharraf))
           (PP (IN with)
               (NP (NP (DT the)
                       (NN task))
                   (PP (IN of)
                       (S (VP (VBG taking)
                              (NP (NN revenge))
                              (PP (IN on)
                                  (NP (NP (DT the)
                                          (NN border)
                                          (NNS tribes))
                                      (, ,)
                                      (ADVP (RB especially))
                                      (NP (DT the)
                                          (ADJP (JJ valiant)
```

```
                                        (CC and)
                                        (JJ lofty))
                                   (NNP Pashtun)
                                   (NNS tribes))))
                              (, ,)
                              (SBAR (IN in)
                                    (NN order)
                                    (S (VP (TO to)
                                           (VP (VB contain)
                                               (NP (NP (DT this)
                                                       (JJ popular)
                                                       (NN support))
                                                   (PP (IN for)
                                                       (NP (NNP jihad))))
                                               (PP (IN against)
                                                   (NP (NP (NN revenge)
                                                           (POS 's))
                                                       (NNP crusader)
                                                       (NN
campaign))))))))))))
                    (. .)))
Therefore     -      (AM-DIS*AM-DIS)     *      *
,      -      *      *      *
America       -      (A0*A0)      (A0*A0)      *
commissioned  commission   (V*V)  *      *
Musharraf     -      (A1*A1)      *      *
with   -      (AM-MNR*     *      *
the    -      *      *      *
task   -      *      *      *
of     -      *      *      *
taking take   *      (V*V)  *
```

```
revenge       -      *      (A1*A1)       *
on      -      *      (AM-LOC*      *
the     -      *      *      *
border  -      *      *      *
tribes  -      *      *      *
,       -      *      *      *
especially    -      *      *      *
the     -      *      *      *
valiant       -      *      *      *
and     -      *      *      *
lofty   -      *      *      *
Pashtun       -      *      *      *
tribes  -      *      *AM-LOC)      *
,       -      *      *      *
in      -      *      (AM-PNC*      *
order   -      *      *      *
to      -      *      *      *
contain       contain       *      *      (V*V)
this    -      *      *      (A1*
popular       -      *      *      *
support       -      *      *      *
for     -      *      *      *
jihad   -      *      *      *A1)
against       -      *      *      (AM-MNR*
revenge       -      *      *      *
's      -      *      *      *
crusader      -      *      *      *
campaign      -      *AM-MNR)      *AM-PNC)      *AM-MNR)
.       -      *      *      *
```

```
(S1 (S (IN So)
       (NP (NN Musharraf))
       (VP (VBD began)
           (S (VP (VP (VBG demolishing)
                      (NP (NNS homes)))
                  (, ,)
                  (VP (VBG making)
                      (NP (NNS arrests)))
                  (, ,)
                  (CC and)
                  (VP (VBG killing)
                      (NP (NP (JJ innocent)
                              (NNS people))
                          (PP (IN in)
                              (NP (NNS markets))))))))
       (. .)))
```

So	-	(AM-DIS*AM-DIS)	*	*	*	
Musharraf	-	(A1*A1)	(A0*A0)	(A0*A0)	(A0*A0)	
began	begin	(V*V)	*	*	*	
demolishing	demolish		(C-A1*	(V*V)	*	*
homes	-	*	(A1*A1)	*	*	
,	-	*	*	*	*	
making	make	*	*	(V*V)	*	
arrests	-	*	*	(A1*A1)	*	
,	-	*	*	*	*	
and	-	*	*	*	*	
killing	kill	*	*	*	(V*V)	
innocent	-	*	*	*	(A1*	
people	-	*	*	*	*	
in	-	*	*	*	*	

```
markets       -      *C-A1) *      *      *A1)
.      -      *      *      *

(S1 (S (NP (NNP Musharraf))
       (, ,)
       (ADVP (RB however))
       (, ,)
       (VP (VBZ pretends)
           (S (VP (TO to)
                  (VP (VB forget)
                      (SBAR (IN that)
                            (S (NP (NP (DT these)
                                       (NNS tribes))
                                   (, ,)
                                   (SBAR (WHNP (WDT which))
                                         (S (VP (AUX have)
                                                (VP (VBN defended)
                                                    (NP (NNP Islam))
                                                    (PP (IN throughout)
                                                        (NP (NP (NNP
Musharraf)
                                                                (POS 's))
                                                            (NN
history)))))))
                                   (, ,))
                               (VP (MD will)
                                   (RB not)
                                   (VP (VB bow)
                                       (PP (TO to)
                                           (NP (DT any)
                                               (JJ treasonous)
```

```
                                                    (NNP US)
                                                    (NN slave))))))))))))
       (. .)))
Musharraf     -      (A0*A0)       (A0*A0)       *      *
,      -      *      *      *      *
however       -      (AM-DIS*AM-DIS)      *      *      *
,      -      *      *      *      *
pretends      pretend       (V*V)  *      *      *
to     -      (A1*   *      *      *
forget forget *      (V*V)  *      *
that   -      *      (A1*   *      *
these  -      *      *      (A0*   (A0*
tribes -      *      *      *A0)   *
,      -      *      *      *      *
which  -      *      *      (R-A0*R-A0)   *
have   -      *      *      *      *
defended      defend *      *      (V*V)  *
Islam  -      *      *      (A1*A1)       *
throughout    -      *      *      (AM-LOC*      *
Musharraf     -      *      *      *      *
's     -      *      *      *      *
history       -      *      *      *AM-LOC)      *
,      -      *      *      *      *A0)
will   -      *      *      *      (AM-MOD*AM-MOD)
not    -      *      *      *      (AM-NEG*AM-NEG)
bow    bow    *      *      *      (V*V)
to     -      *      *      *      (A1*
US     -      *A1)   *A1)   *      *A1)
```

5.4 Triplet Extraction

Our triplet extraction algorithm processes SRL output. SRL splits the input paragraph into sentences and processes entire paragraph sentence by sentence, as seen in the above SRL output . The SRL output format for each input sentence, consist of two components; first a annotated parse tree representation of the input sentence and second multicolumn format representation. We focus on multicolumn format representation to extract triplets, as it consist of a specific pattern, with verb and its corresponding subject, object and preposition attributes in one column. Each column represents one verb (predicate) and its arguments (A0, A1, R-A1, A2, etc) potentially forming triplets.

For a simple sentence, we can read one column and extract a triplet. For complex sentences with many verbs, we developed a bottom-up extraction algorithm for detecting and tagging nested events.

5.4.2. Triplet Matching Rules.

We list four matching rules below to turn simple SRL columns into triplets:

1. A0, V, A1: <SUBJECT, VERB, DIRECT OBJECT>
2. A0, V, A2: <SUBJECT, VERB, PREPOSITION>, if direct object A1 not present in column.
3. A0, V, A1, A2-AM-LOC: <SUBJECT, VERB, DIRECT OBJECT, location (PREPOSITION)>
4. A1, V, A2: <DIRECT OBJ, VERB, PREPOSITION>

Input to Triplet Extraction: Above SRL output in section 5.3.3

Output of Triplet Extraction Algorithm:

Event	Subject	Verb	Object
	America	commission	Musharraf
	America	Take	Revenge
	Musharraf	demolish	Homes
	Musharraf	Make	Arrests
	Musharraf	Kill	innocent people
	Musharraf	Pretend	E1
E1	Musharraf	Forget	E2
E2	tribes	Defend	Islam
E2	tribes	not bow	to US

Table 3: Extracted triplets for the example in section 5.3.3

5.4.1 Bottom-Up Event Tagging Approach

In the example above, consider the triplet <Musharraf, pretend, E1>. Here the object column of the verb pretend has an A1 argument including three other verbs (forget, defend and bow). That is, argument A1 is itself complex, comprising other triplets. So we tag argument A1 with a nested event (E1), and recursively process A1 with our triplet extraction rules. We achieve this nested processing through a bottom-up algorithm that

(i) detects simple verb occurrences (i:e: verbs with non-verb arguments) in the SRL parse tree,

(ii) extracts triplets for those simple verb occurrences using the following Triplet Matching Rules (section 5.4.2),

(iii) replaces simple verb clauses with an event identifier, thus turning all complex verb occurrences into simple verb occurrences with either non-verb or event arguments, and applies the following Triplet Matching Rules.

5.4.3. Triplet Extraction Accuracy

The triplet extraction accuracy is based on SRL accuracy. SRL has precision of 82.28%, recall of 76.78% and f-measure 79.44% [28].

5.4.4. Triplet Based Feature Formation

Above sections (5.4.1 to 5.4.3) explains extraction of triplets in the form <Subject, Verb, Object>. In this section, we discuss formation of triplet features based on these triplets. For each verb (V) mentioned in a story (S), or non-story (NS) we stemmed and aggregated its arguments corresponding to its SUBJECTs, OBJECTs and PREPOSITIONs to generate following set-valued “semantic verb features” by using the training data:

- Argument list for S.V.Subjects, S.V.Objects, S.V.Prepositions for each verb V and story S.
- Argument list for NS.V.Subjects, NS.V.Objects, NS.V.Prepositions for each verb V and Non-Story NS.

For each test paragraph P, for each verb V in P, we extract its typed argument lists P.V.Subjects, P.V.Objects and P.V.Prepositions. Then, we match them to the argument lists of the same verb V. A match succeeds if the overlap between a feature's argument list (e.g. S.V.Subjects, or NS.V.Subjects) covers the majority of the test paragraph's corresponding verb argument list (e.g. P.V.Subjects).

5.5 PropBank Tagging

Proposition Bank [30] (Propbank) is a corpus that is annotated with verbal propositions and their arguments. The need of propbank is that it's difficult to define a general set of semantic roles for all types of predicates (verbs). PropBank defines semantic roles for each verb and sense in the frame files.

Propbank defines semantic roles verb-by-verb basis because of difficulty of defining a universal set of semantic and thematic roles covering all types of predicates. The arguments of a verb are labeled with numbers starting with zero. For example, A0, A1, A2, etc. Propbank defines frame files for each verb which consist of different senses of each of that verb, its arguments (A0, A1, etc) and example of that verb. Frame files are xml files having structure as below:

Frame file 1: hit.xml

hit.01 ; Verb Sense 01 = "strike" ; verb sense id="18.1 18.4"

A0: agent, hitter;

A1: thing hit;

A2: instrument, thing hit by or with

Example - Bank of New England has been hit hard by the region's real-estate slump.

hit.02 ; Verb Sense 02=“reach, encounter”; Verb sense id =”51.8”

A0: thing hitting / reaching

A1 : thing hit / destination

Example-The plane could hit the market in the mid-1990s.

Frame file 2: teach.xml

teach.01 ; Verb Sense 01 = "(try to) make learn” ; verb sense id="37.1-1"

A0: Teacher;

A1: Subject / Topic;

A2: Recipient

Example - Dr. Davulcu teaches Database Management course to computer science students in ASU.

There are 5389 total frame files. We have processed these files and populate the database with primary key as "verb" and other columns are attributes of the verb. So every time instead of processing each file for attribute tags, we can simply read from database and reduce the IO overhead cost.

Propbank tags are used in generalization of triplet features which is explained in next chapter “Generalized Verb Features”.

Chapter 6

GENERALIZED VERB FEATURES

Generalization and reduction of features is an important step in classification process. Extraction of more features requires greater amount of storage space and larger amount feature extraction time. Reduced feature representations not only reduce computing time but they may also yield to better discriminatory behavior. Owing to the generic nature of the curse of dimensionality it has to be assumed that feature reduction techniques are likely to improve classification algorithm.

6.1 Dimensionality Curse

The phenomena, curse of dimensionality, comes in picture when data to be analyzed is in high dimensional spaces - generally hundreds or thousands of dimensions. This phenomena does not occur in low dimensionality spaces such as 2 dimensions or 3 dimensions. This is because as we keep increasing the dimensions, data become sparse. Curse of dimensionality is defined differently in different domain.

For example, in search (nearest neighbor search) domain, assume we have a query data point and search space. In two dimensional Euclidean space, a query with range r forms a circle with area $A = \pi r^2$. But if we increase the dimension to 3 dimensional space, the euclidean space becomes sphere with volume 4/3* πr^3 . As we keep increasing the dimensions to n-dimensional Euclidean space, the volume covered by the query becomes cr^n . Hence, as we keep increasing dimensions, the distance between two data points (query data point and result data point) increases exponentially. Hence, in a n-dimensional vector space, given the data and query distribution, the nearest and furthest

point converges as the dimensionality increases. This is also mentioned that, distance function loses its meaning and effectiveness in high dimensionality.[35] [31]

In machine learning domain, curse of dimensionality is explained below. Large number of features increase noise of the data and thus introduces error in learning algorithm particularly if training data is very limited. This is also referred as Hughes effect [36] which states that with fixed number of training samples, predictive power reduces as dimensionality increases.

Curse of dimensionality is handled by reducing number of features extracted, using generalization approach. In generalization process, features, that are almost identical to each other, are put under single general category. In our classifier, features are based on verbs. To generalize verb based features, verbs that are synonyms with each other are put in a one category and thus reduced number of verb features. Verbnet, a lexical resource which organizes English verbs in different categories, is used to reduce verb based triplet features.

6.2 Verb feature analysis

Our training data had 750 and 1,754 distinct verbs in stories and non-stories, yielding 750 * 3 = 2,250 and, 1,754 * 3 = 5,262 verb features for stories and non-stories respectively, and total of 7,512 features.

VerbNet (VN) [29] is the largest online verb lexicon currently available for English. It is a hierarchical domain-independent, broad-coverage verb lexicon. VerbNet index has 5,879 total verbs represented, and these verbs are mapped into 270 total VerbNet main classes.

For example, the verbs mingle, meld, blend, combine, decoct, add, connect all share the same meaning (i.e. to bring together or combine), and hence they map to verb class “mix” numbered 22.1. With the help of VerbNet and SRL argument types of the verbs, we mapped all occurrences of our verbs in stories and non-stories to one of these 270 VerbNet main classes.

This mapping enabled us to reduce our verb features to 270*6 = 1,620 verb features. The number 6 is used in the preceding equation since each verb class can lead to at most 6 features as V.Subject, V.Object and V.Preposition for its story and non-story occurrences. We started with 7,512 verb features, and after mapping these verb features to their verb category features we ended up with 1,620 features only.

In the generalization process, we faced a problem of verb sense disambiguation. There are some verbs which can be mapped to different senses, and each sense belongs to a different verb class. For example, the verb “add” can be used with the sense ‘mix’ (22.1) or ‘categorize’ (29.2) or ‘say’ (25.3). To solve this problem, we used argument types extracted using SRL for the ambiguous verbs. Then, we performed a lookup for each verb in the PropBank database to identify the matching verb sense with same type of arguments, and its verb class. PropBank [30] is a corpus that is annotated with verbal propositions, and their arguments - a “proposition bank”. In the lookup process, there is a chance that we may encounter more than one verb sense for the input verb matching the corresponding argument types. In this case, we picked the first matching verb sense listed in PropBank.

Chapter 7

EXPERIMENTAL EVALUATION

In this section, we evaluate the utility of standard keyword based features, statistical features based on shallow-parsing (such as density of POS tags and named entities), and a new set of semantic features to develop a story classifier. Feature extraction and matching is implemented using JAVA and classification is performed using LIBSVM [22] in MATLAB.

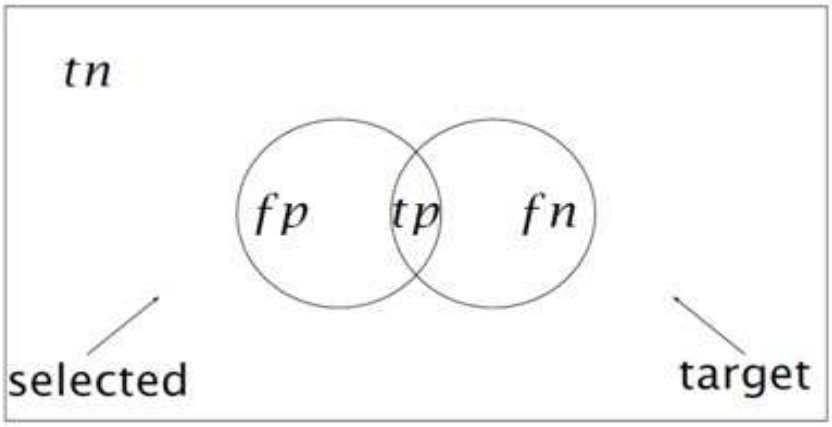

Figure 6: Precision and Recall Venn diagram

7.1 Precision, Recall and F - measure

Figure 6 shows Venn diagram corresponding to the evaluation criteria for classification task. Terminology of above figure is explained below. We define our evaluation criteria as Precision, Recall and F-Measure corresponding to below terminology.

True Positive (tp) - If a test correctly reports the positive result then it's termed as true positive. For example, If a medical test reports a patient as a deceased and in actual he is deceased then the test is true positive.

True Negative (tn) - If a test correctly reports the negative result then it's termed as true negative. For example, If a medical test reports a patient as a healthy and in actual he is healthy then the test is true negative.

False Positive (fp) - If a test falsely or incorrectly report positive result, then its termed as false positive. For example, if a medical test reports a patient with positive result indicating patient is deceased even though he is not deceased in actual, then this medical test is termed as false positive.

False Negative (fn) - If a test falsely or incorrectly reports negative result then the test is termed as false negative. For example, if a medical test reports a patient with negative result indicating patient is not deceased even though patient is deceased, then this medical test is termed as false negative

Predicted Label (SVM classified label)	**Actual Label (Database)**	
Story	Story	True Positive
Story	Non-Story	False Positive
Non-Story	Non-Story	True Negative
Non-Story	Story	False Negative

Table 4 : confusion matrix

Precision (P) : Precision is defined as a measure of the proportion of selected items that system got right. For example, in classifier case, precision states that out of classified

results by SVM, how many are correctly classified. The value of p varies from 0 to 1. The value p=0 states that out of total classified results by SVM, all results are wrongly classified whereas p=1 indicates out of total classified results all results are correctly classified.

$$P = \frac{tp}{(tp+fp)}$$

Recall (R): Recall is defined as the proportion of the target items that the system selected. For example, in our classifier domain, recall means out of total input paragraphs, how many paragraphs are correctly classified. Recall varies from 0 to 1. Recall of 0 means none of not a single paragraph is labeled correctly whereas R=1 states that all input paragraphs are classified by the SVM classifier correctly..

$$R = \frac{tp}{(tp+fn)}$$

F- measure : F- measure is harmonic mean of precision (P) and recall (R) and is a measure of accuracy. F-measure varies between 0 and 1 where 0 is the worst score and 1 is the best score

$$F = \frac{2 (P * R)}{(P + R)}$$

7.2 Results

Feature Set	Precision	Recall	F-measure
POS	0.133	0.066	0.088
Keyword	0.821	0.205	0.329
Keyword + POS + NE	0.750	0.214	0.333
Triplet	0.798	0.515	0.626
Triplet + POS + NE	0.731	0.559	0.634

Table 5. Classifier Performance for Stories

Feature Set	Precision	Recall	F-measure
POS	0.887	0.944	0.914
Keyword	0.903	0.994	0.946
Keyword + POS + NE	0.904	0.991	0.945
Triplet	0.886	0.998	0.938
Triplet + POS + NE	0.905	0.939	0.923

Table 6. Classifier Performance for Non-Stories

7.3 Analysis

The baseline performance for a dummy classifier which would assign all instances to the majority class (non-story) would achieve 80.5% precision and 100% recall for the non-story category; however, its precision and recall would be null for the

stories. Hence, not useful at all for detecting stories. Our proposed model makes use of triplets to incorporate both semantic and structural information available in stories and non-stories.

In Table (5), we report the performance of SVM classification with various feature sets. SVM with POS, NE and generalized triplet based features outperforms other combinations of standard categories of features in terms of recall and F-measure. Table (5) shows 151.2% boost in recall and 90% boost in F-measure for keyword based (second row) vs. triplet based (fourth row) features. After adding POS and NE features (Keyword + POS + NE based, third row) vs. (Triplets + POS + NE, fifth row), we obtained 161.2% boost in recall and 90% boost in F-measure.

Chapter 8

CONCLUSION AND FUTURE WORK

This paper proposes a model with semantic triplet based features for story classification. The effectiveness of the model is demonstrated against other traditional features used in the literature for text classification tasks.

Future work includes more detailed evaluations, and also experiments with appropriate generalizations of nouns, adjectives and other types of keywords found in verb arguments.

REFERENCES

[1] J. Bruner and S. Weisser, “Autobiography and the construction of self,” 1992.

[2] C. Joseph, The hero with a thousand faces. Princeton University Press, 1949.

[3] H. L. Halverson, J. R. Goodall and S. R. Corman, Master Narratives of Islamist Extremism. New York: Palgrave Macmillan, 2011.

[4] R. Hoffman, U. Grasemann, R. Gueorguieva, D. Quinlan, D. Lane, and R. Miikkulainen, “Using computational patients to evaluate illness mechanisms in schizophrenia,” Biological psychiatry, 2011.

[5] B. Masand, G. Linoff, and D. Waltz, “Classifying news stories using memory based reasoning,” in Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 1992, pp. 59–65.

[6] D. Billsus and M. Pazzani, “A hybrid user model for news story classification,” Lectures-International Centre for Mechanical Sciences, pp. 99–108, 1999.

[7] A. S. Gordon and K. Ganesan, “Automated story capture from conversational speech,” in K-CAP ’05: Proceedings of the 3rd international conference on Knowledge capture, ACM. Banff, Canada: ACM, 2005, p. 145–152.

[8] A. Gordon, Q. Cao, and R. Swanson, “Automated story capture from internet weblogs,” in Proceedings of the 4th international conference on Knowledge capture. ACM, 2007, pp. 167–168.

[9] A. Gordon and R. Swanson, “Identifying personal stories in millions of weblog entries,” in Third International Conference on Weblogs and Social Media, Data Challenge Workshop, San Jose, CA, 2009.

[10] J. Fleiss, “Measuring nominal scale agreement among many raters.” Psychological bulletin, vol. 76, no. 5, p. 378, 1971.

[11] E. F. Tjong Kim Sang and F. De Meulder, “Introduction to the conll-2003 shared task: Language-independent named entity recognition,” in Proceedings of CoNLL- 2003, W. Daelemans and M. Osborne, Eds. Edmonton, Canada, 2003, pp. 142–147.

[12] T. Poibeau and L. Kosseim, “Proper name extraction from non-journalistic texts,” Language and Computers, vol. 37, no. 1, pp. 144–157, 2001.

[13] J. Finkel, T. Grenager, and C. Manning, “Incorporating non-local information into information extraction systems by gibbs sampling,” in Proceedings of the 43rd Annual

Meeting on Association for Computational Linguistics. Association for Computational Linguistics, 2005, pp. 363–370.

[14] L. Ratinov and D. Roth, “Design challenges and misconceptions in named entity recognition,” in CoNLL’09: Proceedings of the Thirteenth Conference on Computational Natural Language Learning. Morristown, NJ, USA: Association for Computational Linguistics, 2009, pp. 147–155.

[15] “Calais.” [Online]. Available: http://www.opencalais. com/

[16] S. Auer, C. Bizer, G. Kobilarov, J. Lehmann, and Z. Ives, “Dbpedia: A nucleus for a web of open data,” in 6th Intl Semantic Web Conference, Busan, Korea. Springer, 2007, pp. 11–15.

[17] S. Robertson, “Understanding inverse document frequency : on theoretical arguments for idf,” Journal of Documentation, vol. 60, no. 5, pp. 503–520, 2004.

[18] E. Brill, “A simple rule-based part of speech tagger,” in Proceedings of the workshop on Speech and Natural Language. Association for Computational Linguistics, 1992, pp. 112–116.

[19] T. Joachims, “A statistical learning model of text classification for support vector machines,” in Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2001, pp. 128–136.

[20] B. Boser, I. Guyon, and V. Vapnik, “A training algorithm for optimal margin classifiers,” in Proceedings of the fifth annual workshop on Computational learning theory. ACM, 1992, pp. 144–152.

[21] C. Cortes and V. Vapnik, “Support-vector networks,” Machine learning, vol. 20, no. 3, pp. 273–297, 1995.

[22] C. Chang and C. Lin, “Libsvm: a library for support vector machines,” ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, p. 27, 2011.

[23] S. Keerthi and C. Lin, “Asymptotic behaviors of support vector machines with gaussian kernel,” Neural computation, vol. 15, no. 7, pp. 1667–1689, 2003.

[24] D. Rusu, L. Dali, B. Fortuna, M. Grobelnik, and D. Mladeni´c, “Triplet extraction from sentences,” Proceedings of the 10th International Multiconference Information Society-IS, pp. 8–12, 2007.

[25] D. Hooge Jr, “Extraction and indexing of triplet-based knowledge using natural language processing,” Ph.D. dissertation, University of Georgia, 2007.

[26] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky, “Stanfords multi-pass sieve coreference resolution system at the conll-2011 shared task,” CoNLL 2011, p. 28, 2011.

[27] K. Raghunathan, H. Lee, S. Rangarajan, N. Chambers, M. Surdeanu, D. Jurafsky, and C. Manning, “A multipass sieve for coreference resolution,” in Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 2010, pp. 492–501.

[28] V. Punyakanok, D. Roth, and W. Yih, “The importance of syntactic parsing and inference in semantic role labeling,” Computational Linguistics, vol. 34, no. 2, pp. 257–287, 2008.

[29] K. Kipper, A. Korhonen, N. Ryant, and M. Palmer, “A large-scale classification of english verbs,” Language Resources and Evaluation, vol. 42, no. 1, pp. 21–40, 2008.

[30] M. Palmer, D. Gildea, and P. Kingsbury, “The proposition bank: An annotated corpus of semantic roles,” Computational Linguistics, vol. 31, no. 1, pp. 71–106, 2005.

[31] K. Selcuk Candan, Maria Luisa Sapino, "Data Management for Multimedia Retrieval", Cambridge university press, 2010.

[32] LIBSVM, http://www.csie.ntu.edu.tw/~cjlin/libsvm/oldfiles/index-1.0.html.

[33] Cross Validation, http://en.wikipedia.org/wiki/Cross-validation_(statistics)

[34] Dali, Lorand, and Blaz Fortuna. "Triplet extraction from sentences using svm."Proceedings of SiKDD (2008).

[35] Beyer, K.; Goldstein, J.; Ramakrishnan, R.; Shaft, U. (1999). "When is “Nearest Neighbor” Meaningful?". Proc. 7th International Conference on Database Theory - ICDT'99. LNCS 1540: 217–235.

[36] Alonso, María C., José A. Malpica, and A. M. Agirre. "Consequences of the hughes phenomenon on some classification Techniques." ASPRS 2011 Annual Conference, Milwaukee, Wisconsin May. 2011.