Unsupervised Learning for Information Distillation

Kamand Kamangar [1]

IDIAP–RR 07-47

October 30, 2007

[1] Ecole Polytechnique Fédérale de Lausanne. IDIAP Research Institute, International Computer Science Institute, Berkeley, CA Speech Group, `kamand@ICSI.berkeley.EDU`

IDIAP Research Report 07-47

Unsupervised Learning for Information Distillation

Kamand Kamangar

October 30, 2007

Abstract.

Current document archives are enormously large and constantly increasing and that makes it practically impossible to make use of them efficiently. To analyze and interpret large volumes of speech and text of these archives in multiple languages and produce structured information of interest to its user, information distillation techniques are used. In order to access the key information in response to a request (query), special text processing techniques such as distillation are required. The task consists of filtering methods to extract the important portions of relevant documents, named as snippets, to a query, as concisely and as correctly as possible. In the context of GALE Project, the queries are matched into several predefined templates. Template 1, which is the main focus of this work, corresponds to listing of facts about an EVENT. Answering to template 1 questions is much similar to extraction of general passages from an IR engine. In this work, we are implementing an iterative unsupervised method to answer the queries of this template. The goal of unsupervised learning is to increase the performance of classifier in terms of error rate and f-measure without depending to prior annotated data. The approach consists of using only highly confident features such as word transcriptions extracted from query as well as their synonyms. After forming the bootstrap model using these features, the model is improved using self-training, and is iteratively trained during consecutive runs. Our results indicate that the performance of the system may be improved by more than 12.5% relative in terms of classification error and 31% relative in terms of F-measure, by using the proposed methods.

Acknowledgments

My experience at ICSI was instructive, challenging, and productive. I would like to acknowledge those who made it possible for me to have this great experience.

Dr. Dilek Hakkani-Tür, my advisor at ICSI, never stopped helping and advising me, even in the hardest of times, for which I am truly indebted to her. Working with Dr. Hakkani-Tur will always remain a truly memorable experience for me.

Prof. Hervé Bourlard, my advisor at EPFL, taught me how to plan for my thesis, encouraged me to apply for this internship at ICSI, helped me through the application process, and showed kind trust and support during my thesis project.

Prof. Nelson Morgan kindly provided me with the opportunity to work at ICSI.

Dr. Gokhan Tur provided me with very instructive and helpful comments on analysis and interpretation of the results.

I received kind advice and support from Dr. Michael Levit, my colleague at ICSI, on technical parts on work during the intense period of writing this report.

Sébastien Cuendet, my colleague at EPFL, helped me learn the concepts of classification and initiate my thesis project.

Again, I would like to express my deep appreciation and gratitude toward the individuals listed above. I enjoyed my time at ICSI and learned a lot during my thesis project. Thank you, all.

Abstract

Current document archives are enormously large and constantly increasing and that makes it practically impossible to make use of them efficiently. To analyze and interpret large volumes of speech and text of these archives in multiple languages and produce structured information of interest to its user, information distillation techniques are used. In order to access the key information in response to a request (query), special text processing techniques such as distillation are required. The task consists of filtering methods to extract the important portions of relevant documents, named as snippets, to a query, as concisely and as correctly as possible.

In the context of GALE Project, the queries are matched into several predefined templates. Template 1, which is the main focus of this work, corresponds to listing of facts about an [EVENT]. Answering to template 1 questions is much similar to extraction of general passages from an IR engine.

In this work, we are implementing an iterative unsupervised method to answer the queries of this template. The goal of unsupervised learning is to increase the performance of classifier in terms of error rate and f-measure without depending to prior annotated data. The approach consists of using only highly confident features such as word transcriptions extracted from query as well as their synonyms. After forming the bootstrap model using these features, the model is improved using self-training, and is iteratively trained during consecutive runs. Our results indicate that the performance of the system may be improved by more than 17% relative in terms of classification error and 30% relative in terms of F-measure, by using the proposed methods.

Keywords: distillation, template-based question answering, unsupervised classification, boosting, lexical features.

Contents

CHAPTER
1

Introduction

Documents are one of the most important repository sources of information which are produced from textual or audio resources. Textual document are formed primarily by news feeds or blogs. Documents pertain to large amounts of data, about an *event* at a certain time period. Important documents are usually held in large archives to be accessed later for answering critical questions, in appropriate situations. The oversize of current archives, which increase constantly, makes it practically impossible to make use of them efficiently. Therefore, in order to access the key documents in response to a request(query), special text processing techniques are required. The task consists of filtering methods to extract the important portions of relevant documents as concisely and as correctly as possible. In the framework of the DARPA Global Autonomous Language Exploitation (GALE) program, this is named as distillation. Distillation is useful for direct access to relevant information of interest in the corpus. Output or information of interest consists of useful pieces of information to a given query, named *snippets*, possibly from a multilingual audio and text corpus (Hakkani-Tür & Tur, 2007).

In GALE Project, the queries are matched into several predefined templates. Template 1, which is the main focus of this work, corresponds to the extraction of *list of facts about an [EVENT]*. Data pool to answer the query consists of a collection of documents which comes from an information retrieval (IR) Engine. Once the documents are returned, the automated response generators may be used to answer the query. Several methods have been developed to produce confident responses to a question such as statistical methods for sentence extraction. In these methods, to train the sentence extraction models, the negative and positive examples from the *given answer keys* (annotated data) are extracted (Hakkani-Tür *et al.*, 2007). Other approaches have tried to manually write rules or patterns, for extracting sentences that may have a relevant response to a query as well.

1.1. Related Work To Distillation

The purpose of distillation is to output ordered segments of data called snippets. Snippets can be considered as an answer to a query. A snippet can range from a fragment of a sentence to a paragraph.(Hakkani-Tür *et al.*, 2007). The distillation task is similar in nature to question answering task, however, is not the exactly same (Voorhees, 2003).

One of the significant approaches to distillation is the IXIR system which is a common work of ICSI and SRI. In this work, basically the relevant documents are extracted by document retrieval. DR is defined as the matching of a stated query against useful parts of free-text records. These records could be any type of unstructured text, such as newspaper articles.

A document retrieval system has two main tasks:

- Find *relevant documents* to user queries,
- *Evaluate the matching results* and sort them according to relevance.

Document classification tasks can be divided into two main sorts: supervised document classification where some external mechanism (such as human feedback) provides information on the correct classification for documents, and unsupervised document classification, where the classification must be done entirely without reference to external information. There is also a third sort called semi-supervised classification, but we don't use or discuss about this term in this work.

In IXIR approach, inside these relevant documents, relevant sentences are extracted via classification. To extract relevant documents, document retrieval process is based on Information Retrieval(IR) and Information Extraction(IE) (Levit *et al.*, 2007a). Relevant documents are found with IR, and are constrained with IE annotations related to the query type.

Information Retrieval (IR) is the science of searching for information in documents for a given query, searching for documents themselves, searching for meta-data which describe documents, or searching within databases and extracting the sentences/snippets describing the event in the query with associated entities.

Information Extraction (IE) serves to extract all the named entities, co-references (mentions), relationships and *events in the document sources*. These IE elements are preferred to be extracted from ACE (Automatic Content Extraction) annotation guidelines, defined by the annual NIST ACE evaluations. In next chapters, we explain more in detail about ACE annotations.

In the framework of GALE project, it has been shown that a combination of Information Extraction and Information Retrieval leads into better results in terms of Recall and Precision (Hakkani-Tür *et al.*, 2007).

In such kind of learning methods, the classifier is usually trained by *a large amount of annotated data* in corpus. However, annotated data may not be available for particular query and it is time consuming to label them manually. In addition, labeling data by humans has a high risk of error. Thus, the need for a model which does not require prior labeled data is evident.

An efficient classifier can be a solution to this problem. If the classifier automates sentence annotating, it can reduce enormously the manpower and this is feasible by an unsupervised distillation method. Full or partial sentence annotating can be later used for supervised or semi-supervised classification, respectively.

Besides, template 1 queries are very general questions with a wide domain of responses. Answering to these kind of questions is much similar to extraction of general passages from an IR engine. In other words, the meaning of template is not really applied in this case.

Having this motivation, we decided to implement an unsupervised method to answer the queries of template 1. The approach consists of using only high confidence examples selected by help of word transcriptions extracted from Q_i plus their synonyms.

Since we assume no training data is available, we bootstrap a classifier using examples that are relevant to the query with a high confidence (such as sentences that have all the words in the query slot) and those that are irrelevant to the query (such as sentences that do not contain any of the words in the query slot. Then we use this bootstrap classifier to label all sentences in the documents that may be relevant to the query. We iterate this process until the classification performance on the held-out set converges, or the estimated classes converge. At each iteration, we re-train the classifier using examples that are classified with a high confidence by the classifier of the previous iteration.

1.2. Structure of this work

Next chapter is devoted to explain the concept of different types of machine learning. Then we explain the classification concept, a particular usage of machine learning, which will be the pivot of this project. Throughout this work, we focus and use only one machine learning algorithm, the Boosting algorithm. The arithmetic base of this algorithm will be explained in order to understand the further results we obtain in our experiments.

Chapter 3 discusses the practical description of distillation process. Then we talk about the GALE program and its text processing engines. This part will be followed by some examples of template based query answering and document triage. In the last part of the chapter, related work to distillation technology and improvements in this area will be covered.

In chapter 4, we explain our approach in testing the AdaBoost classifier from the very beginning until the development of a complete unsupervised learning algorithm. We talk about the methods that we tried to achieve the best set of results.

Chapter 5 describes the data sets and the metrics that are particularly used in the experiments.

In chapter 6, we present the results of the various experiments that were performed and discuss about the reasons for success or failure of the approaches.

Finally, Chapter 7 summarizes the results and deduction of current work along with the future challenges in the area of non supervised template based query answering.

CHAPTER
2

Classification

This chapter describes the global term of Machine Learning and proceeds in particular with the concept of classification. Then we explain some important terms required to understand the subject thoroughly. Finally, we talk about a successful classification method and the classifier tool of AT&T, AdaBoost, which performs this task. The arithmetic base of this algorithm will be explained for a better understanding of the concept.

2.1. Machine Learning

Machine learning refers to a system capable of the autonomous acquiring and integration of knowledge (Freitag, 2000). The ultimate goal of such as system is to develop more broadly applicable systems with more robust learning capabilities. Successful, machine learning research could produce computer systems that learn to operate in novel environments like speech understanding systems that automatically adapt to new speakers and new environmental conditions. They may act as knowledge based consultant systems that collaborate with human experts to solve difficult problems and acquire new problem-solving tactics by observing the human's contribution to the eventual problem solution. The goal of machine learning research is to produce a *domain-independent* enabling technology for a broad range of computer applications. Many applications of computers are increasingly knowledge based, i.e. dependent on a large number of specific facts about the task domain.

This capacity to learn from experience and analytical observation, results in a system that can continuously self-improve and consequently offer increased efficiency. Machine learning offers the potential to remove the knowledge acquisition bottleneck that limits performance and increases development costs for such systems (Sequel, 2007).

Machine learning approaches can be divided into 3 main categories:

2.1.1 Supervised Learning

Supervised Learning is a machine learning technique for creating a function from training data. The training data consist of pairs of input objects (typically vectors), and desired outputs. The output of the function can be a continuous value (called regression), or can predict a class label of the input object. The task of the supervised learner is to predict the value of the function for any valid input object after having seen a number of training examples (i.e. pairs of input and target output). To achieve this, the learner has to generalize from the presented data to unseen situations in a "reasonable" way.

2.1.2 Semi-supervised Learning

In computer science, semi-supervised learning is a class of machine learning techniques that make use of both labeled and unlabeled data for training - typically a small amount of labeled data with a large amount of unlabeled data. Semi-supervised learning falls between unsupervised and supervised learning. Machine-learning researches show that unlabeled data, when used in conjunction with a small amount of labeled data, can produce considerable improvement in learning accuracy. The acquisition of labeled data for a learning problem often requires a skilled human agent to manually classify training examples. The cost associated with the labeling process thus may render a fully labeled training set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. In such situations, semi-supervised learning can be of great practical value.

One example of a semi-supervised learning technique is self-training where the learner iteratively classifies unlabeled data and adds the ones which have got high enough confidence to its trainnig data, optionally with a weight. While this method is shown to be effective in a number of speech and language processing tasks, it has the risk of diverging greatly from the optimal learner, so a held out control is needed to measure the relative improvement. Another approach is called co-training, in which two or possibly more learners are each trained on a set of examples, but with each learner using a different, and ideally independent set of features for each example. For example, for web page classification one view may be the words in the web page and another view may be the words in the hypoerlinks pointing to that web page.

An alternative approach is to model the joint probability distribution of the features and the labels. For the unlabeled data the labels can then be treated as 'missing data'. It is common to use the Expectation-Maximization(EM) algorithm for finding maximum likelihood of the model. This estimates the parameters in probabilistic models, where the model depends on unobserved latent variables which are not directly observed but are rather inferred.

2.1.3 Unsupervised Learning

Unsupervised learning is a method of machine learning where a model is built based on observations. It is distinguished from supervised learning by the fact that there is no a priori output. In unsupervised learning, a data set of input objects is gathered. Unsupervised learning then typically treats input objects as a set of random variables. A joint density model is then built for the data set.

Unsupervised learning can be used in conjunction with Bayesian inference to produce conditional probabilities (i.e. supervised learning) for any of the random variables given the others. An important usage of unsupervised learning is for clustering, which is sometimes not probabilistic.

From a theoretical point of view, supervised and unsupervised learning differ only in the causal structure of the model. In supervised learning, the model defines a set of observations, called inputs and has on another set of observations, called outputs. The inputs are assumed to be at the beginning and outputs at the end of the causal chain. The models can include mediating variables between the inputs and outputs.

In unsupervised learning, all the observations are assumed to be caused by latent variables, that is, the observations are assumed to be at the end of the causal chain. In practice, models for supervised learning often leave the probability for inputs undefined. This model is not needed as long as the inputs are available, but if some of the input values are missing, it is not possible to infer anything about the outputs. If the inputs are also modeled, then missing inputs cause no problem since they can be considered latent variables as in unsupervised learning.

2.2. Important Definitions

In the rest of this work, we will use some words that are specific to machine learning and classification. Since different terminologies are used in literature, first we specify the meaning of some expressions that will be used frequently throughout this work.

Classifier. A classifier is the tool or program used for learning process. It is based on a predictive function $c = f(x)$ that, given an instance x and a set of classes C, associates one class $c \in C$ to the instance x, it means that it classifies the instance x.

Labeled or Annotated Data. Labeled(annotated) data refers to a set of data in which each instance in the data set is *manually* assigned a corresponding class. Unlabeled(non annotated) refers to the instances of the data that do not have a class assigned by a human labeler.

Training or Building a Classifier. It consists of applying a learning algorithm $\mathcal{L}$ with some training data D_t results in the creation of a classifier. We call this process the *training* phase. In other terms, the resulting classifier is *built* on D_t using $\mathcal{L}$.

Testing or Evaluating a Classifier. Once a classifier has been trained on some data D_t it can be *evaluated* on another set of data D_e, with $D_t \cap D_e = \varnothing$. In the evaluation phase, also called testing phase, the classifier attributes a class $c_e \in C$ to every instance $x \in D_e$. The class c_e that was attributed by the classifier to the instance x is then compared to the true class c_t of the instance x. If the two classes match, i.e. $c_e = c_t$, the classification of x is considered correct. In the opposite case ($c_e \neq c_t$), the classification of x is considered erroneous. Different metrics can then be used to evaluate the classification performance, based on the number and the type of instances erroneously and correctly classified.

2.3. Classification

The major focus of Machine learning research is to automatically extract information from data by computational and statistical methods. Hence, machine learning is closely related to data mining and statistics.

Classification, as a sub-method of Machine Learning, can also be performed by both supervised and unsupervised algorithms. Supervised learning has the strict meaning of classification where the classes are known a priori and have a meaning to the user. Unsupervised document classification or document clustering has been used to enhance information retrieval. This is based on the clustering hypothesis, which states that documents having similar contents are also relevant to the same query. A fixed collection of text is clustered into groups or clusters that have similar contents. The similarity between documents is usually measured either with boosting algorithms or with the associative coefficients from the vector space model, e.g., the cosine coefficient.

Statistical classification is a statistical procedure in which individual items are placed into groups based on quantitative information on one or more characteristics inherent in the items (referred to as features) and based on a training set of previously labeled items.

The learning algorithm should be appropriate to the task. For example, while support vector machines (SVM) yield top results in most of the learning tasks (Liu *et al.*, 2006), studies have shown that the BoosTexter tool (Schapire & Singer, 2000), which implements a boosting learning algorithm, is particularly efficient for text classification and sentence segmentation problems.

2.4. Boosting, A Successful Classification Method

Boosting is a machine learning meta-algorithm for performing supervised learning. Boosting tries to create a single strong learner from a set of weak learners. A weak learner is defined to be a classifier which is only *slightly correlated* with the true classification. In contrast, a strong learner is a classifier that is arbitrarily *well-correlated* with the true classification. While boosting is not basically a constrained algorithm in general, most boosting algorithms follow a template. Typically boosting occurs in iterations, by incrementally adding weak learners to a final strong learner. At every iteration, a weak learner learns the training data with respect to a distribution. The weak learner is then added to the final strong learner. This is typically done by weighting the weak learner in some manner, which is typically related to the weak learner's accuracy. After the weak learner is added to the final strong learner, the data is reweighed and examples that are misclassified gain weight and examples that are classified correctly lose weight. Thus, future weak learners will focus more on the examples that previous weak learners misclassified (Kearns, 1998). Briefly, the algorithm tries to improve the final result of a learning algorithm by iteratively training a classifier on a weighted training set of data. At the end of the learning process, the weighted linear combination of the classifiers built at each iteration forms the final classifier.

Some algorithms refer to themselves as "boosting algorithms", and these algorithms can be quite effective. However, only provable boosting algorithms should adopt the title "boosting". These algorithms that are similar to boosting are sometimes called "leveraging algorithms".

The main variation between many boosting algorithms is their method of weighting training data points and hypotheses. AdaBoost is very popular and perhaps the most significant historically as it was the first algorithm that could adapt to the weak learners. However, there are many more recent algorithms such as LPBoost, TotalBoost, BrownBoost, MadaBoost, LogitBoost, and others.

2.5. AdaBoost Description

During the last decades, a large number of machine learning methods have been proposed for text classification tasks which typically built around the Bag-of-Words model known from information retrieval. But this approach comes with deficiencies that motivate the integration of features on a higher semantic level than single words(Bloehdorn & Hotho, 2005).

Boosting has its roots in a theoretical framework for studying machine learning called the "PAC" learning model. They tried to enhance a weak learning algorithm which performs just slightly better than random guessing in the PAC model in order to make it boosted into an arbitrarily accurate strong learning algorithm. Boosting is part of a larger theory called ensemble learning. While ordinary machine learning algorithms, such as decision trees or preceptors, work by searching the best hypothesis for a set of data, ensemble learning algorithms have a different approach. Rather than finding a unique hypothesis, they construct a committee of simple hypotheses and combine them to obtain a final hypothesis.

AdaBoost is a well-known algorithm that uses boosting. In this work we use the particular AdaBoost.MH version of the algorithm by AT&T. This is the algorithm which we will refer to as Boosting in the rest of this report and implicitly when mentioning boosting in general.

When AdaBoost was first presented, immediately it drew the attention in the machine learning community, since it was one of the most accurate classification algorithms available at that time(Freund & Schapire, 1995). At the beginning, researchers associated it to bagging and thought that the major strength of Boosting had to do with variance reduction. But experiences showed that Boosting also reduces bias and thus differs from bagging(Friedman *et al.*, 1998). It has since then led to more research, both on the practical side and on the theoretical side and is still a widely and lively discussed topic in the machine learning field. An interesting characteristic of Boosting is that it seems to be resistant to over-fitting. This phenomenon had no theoretical explanation until 1998 when Schapire et al. found an upper bound error for Boosting and argued that the non over-fitting effect was explained by the fact that Boosting produced a high margin distribution. Being an efficient learning algorithm in general, Boosting is quite appropriate for template-based query answering task since it handles word N-grams which is one of the basic ideas for our approach as will be explained later.

2.6. Algorithm

BoosTexter implements different versions of the AdaBoost algorithm for the task of text categorization. The version used in this work has a discriminative power because it not only checks for the presence of a feature but also for the absence of it. As explained above, the idea of boosting is to run several times one simple classifier on a weighted training set of data, changing the weights at each iteration.

Particularly at each iteration t, a new simple classifier h_t is built and all simple classifiers $h_t, t \in [1, \ldots, T]$ are combined at the end of the T iterations of the learning process into one final classifier $f(x,l)$:

$$f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l) \tag{2.1}$$

with α_t the weight of the weak learner h_t.

In the case of boosting, a weak learner has the same form as a one-level decision tree. The pseudocode of the AdaBoost.MH algorithm is shown below.

Each training example i is assigned L weights $D(i,l)$, i.e. one weight for each possible class. At the beginning of the learning process, all weights are equal. At each iteration, a weak learner finds a weak hypothesis $h_t(x,l) : \mathcal{X} \rightarrow \Re$ given the distribution of the weights $D(i,l)$. The sign of $h_t(x,l)$ is interpreted as a prediction of whether the label l belongs to the set of labels assigned to the sample x ($h_t(x,l) > 0$) or not ($h_t(x,l) \leq 0$). Once this evaluation has been done for each label l, the following function $Y[l]$ can be described:

$$Y[l] = \begin{cases} 1, & \text{if } l \in Y \\ 0, & \text{if } l \notin Y \end{cases}$$

The weights are then updated such that the next iteration $t+1$ will force the weak learner to focus on the examples that were wrongly classified at the last iteration t. The final hypothesis is

```
Given the training data: $(x_1, Y_1), ..., (x_m, Y_m)$ where $x_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y}$
Initialize the distribution $D_1(i,l) = 1/mk, i = 1..m, k = 1..|\mathcal{Y}|$
**for** each iteration $t = 1..T$ **do**
    Train a base learner $h_t$ using distribution $D_t$
    Update
        $D_{t+1}(i,l) = \frac{D_t(i,l)e^{(-\alpha Y_i[l]h_t(x_i,l))}}{Z_t}$    (2.2)
    where $Z_t$ is a normalization factor and $\alpha_t$ is the weight of the base learner
**end for**
The final classifier is the weighted sum of the $h_t(x,l)$:
        $f(x,l) = \sum_{t=1}^{T} \alpha_t h_t(x,l)$
```

FIG. 2.1: AdaBoost.MH

a weighted linear combination of the hypotheses of all iterations, with higher weights attributed to the weak hypotheses with a lower classification error rate.

The final output of the boosting algorithm described in the above Algorithm yields a real number value w =f (x, l)for each example x and label l. These values are weights of the label l, attributed to a sample x. To achieve the final goal of classifying, the weights f (x, l)need to be interpreted and mapped to a class. The conversion from the weights output by the Boosting to the attribution of a class is done as described below.

Each weight is turned into probability $p(l|x)$ according to the following equation:

$$p(l|x) = \frac{1}{1 + e^{-2f(x,l)T}} \tag{2.3}$$

where T is the number of iterations used to train the classifier and $f(x,l)$ the final classifier output by the learning algorithm.

These probabilities can then be converted into a binary classification by using a threshold *thr*.

$$H(x) = \begin{cases} 1, & \text{if } p(C_s|x) > thr \\ 0, & \text{if } p(C_s|x) \leq thr \end{cases} \tag{2.4}$$

where $H(x)$ reflects the class of the example x. The threshold *thr* is a parameter and can be optimized during run-time.

2.7. Weak Learner's Error

The weak learner's job is to find a weak hypothesis at each iteration. The weak hypotheses returned by the weak learner throughout the T iterations all have the form of a one-level decision tree. Even though the weak hypothesis $h_t(x_i,l)$ maps each sample x to a real value in $\Re$, it is easier to understand it in the simplest case where each $h_t(x_i,l)$ is binary, i.e. restricted to $\{-1,+1\}$. The base learner minimizes the probability that the wrong label will be attributed to a sample x for the current weights distribution D_t. However, the strength of a weak hypothesis is measured by its error (Freund & Schapire, 1999).

The error function is expressed by the following function:

$$\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq Y_i[l]] \tag{2.5}$$

In practice, the error is measured with respect to the distribution D_t on which the weak learner was trained. The weak learner may be an algorithm that can use the weights D_t on the training examples. Alternatively, when this is not possible, a subset of the training examples can be sampled according to D_t, and these un-weighted re-sampled examples can be used to train the weak learner.

For each example x, the algorithm tests the previous word. The weak learner then chooses the word w among all the preceding words and create a basic rule saying "*if the word in instance x is preceded by the word w than give it value r_1, otherwise give it value r_2*". The choice of the word w takes into account the overall weighting error made by choosing this word. In the usual case where more than one feature is available, the best hypothesis obtained out of all features is returned.

Once the algorithm has been provided with a base classifier h_t, the next step is to determine the weight $\alpha_t \in \Re$ that will be attributed to h_t in the linear combination that forms the final classifier. In the case of a binary hypothesis $h_t(x_i, l)$, this is done as follows:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Thus, if the error ϵ_t inferred by the base classifier h_t is big, the weight α_t assigned to h_t is small. A small α_t results in a higher exponent for e in the weights update (2.2). If the hypothesis done by h_t was correct for the example x_i, y_i and $h_t(x_i)$ have the same sign and the exponent is thus negative, leading to a reduction of the weights $D(i, l)$ of the example x_i. On the contrary, if the hypothesis done by h_t was wrong, the signs of $Y_i[l]$ and $h_t(x_i)$ in 2.5 are opposed, leading to a positive exponent and thus to an increase of the weight $D(i, l)$ of sample x_i. The sample x_i will therefore have a bigger importance in the next iteration when a new base classifier will be chosen.

CHAPTER
3

Distillation

In this chapter, we discuss about the distillation concept and important uses of this technology. Then we talk about Global Autonomous Language Exploitation program (GALE) and its text processing engines. This part will be followed by some examples of template based query answering and document triage. In the last part of the chapter, related work to distillation and question answering technology, and improvements in this area will be explained briefly.

3.1. Distillation

In recent years, technology has progressed quite rapidly, from systems that could accurately process text in only very limited domains like service reports to programs that can perform useful information extraction from a very broad range of texts such as business news. The major strength behind these advances comes from: (1) the development of robust text processing architectures and (2) the emergence of statistical methods that help to overcome knowledge acquisition problems by making use of corpus and training data.

Text processing of large amounts of data requires also special filtering methods to extract the required data as concisely and correctly as possible. One of the most important techniques is distillation. The goal of this process is to develop and apply technologies to analyze and interpret the huge volumes of speech and text in multiple languages. Particularly, it is useful for direct access to relevant information of interest in the corpus. Output or information of interest consist of useful pieces of information to a given query, possibly from a multilingual audio and text corpus (Hakkani-Tür & Tur, 2007). The distillation engine integrates the information from multiple sources and documents and currently its output is English text.

3.2. Global Autonomous Language Exploitation program(GALE)

During this project, we are using the distillation engine of DARPA Global Autonomous Language Exploitation program(GALE). The goal of the GALE program is to develop and apply computer software technologies to translate, analyze, and interpret large volumes of speech and text in multiple languages. This is mainly done for reducing the need for linguists and analysts and manpower in general for automatically providing relevant and concise information. This technology will make it possible for English speakers to do much of what foreign language operators and analysts do now, i.e. convert large quantities of non-decipherable foreign language data into usable data in English for operational planning, crisis response, and force protection.

Automatic processing engines will convert and distill the data, delivering pertinent information in easy-to-understand forms to monolingual English-speaking analysts in response to direct or implicit requests. Critical information, usually buried in large volumes and obscured by foreign languages, is both highly important and subject to quick destruction. The GALE program will provide in an integrated product, automated transcription, translation, and distillation of foreign speech and text in support of military operations and tactical situational awareness. 3.1 illustrates the concept of GALE Distillation engine and consecutive steps of query response generation. The process accepts English or non-English documents along with a query. The following progressive steps are done to generate the query response:

1. Identifying of relevant information
2. Eliminating the redundant information
3. Detecting of contradictions
4. Proposition for an already supported response by some evidence

The output of this system generates a comprehensive query response to the end-user.

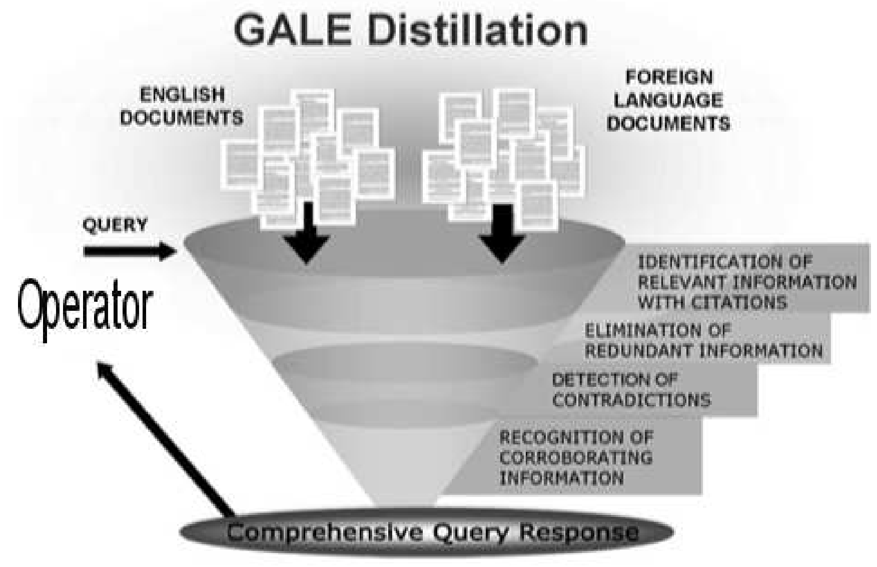

GALE Distillation
ENGLISH DOCUMENTS
FOREIGN LANGUAGE DOCUMENTS
QUERY
Operator
IDENTIFICATION OF RELEVANT INFORMATION WITH CITATIONS
ELIMINATION OF REDUNDANT INFORMATION
DETECTION OF CONTRADICTIONS
RECOGNITION OF CORROBORATING INFORMATION
Comprehensive Query Response

FIG. 3.1: Gale Concept and Goals

GALE utility consists of three integrated parts: transcription, translation, and distillation. 3.2 depicts these three engines and their functionalities. Transcription, or the conversion of audio (speech) to English text, is the first step in exploiting foreign language audio information. Translation is the conversion of foreign language text into readable English text with annotations related to the language of origin, topics, parts of speech, names and other factors and is carried out on text transcribed from audio or obtained from foreign language text sources. GALE engines perform both of these processes in a completely automated fashion, without the intervention of human linguists. The distillation engine will search for and integrate information from multiple sources relevant to specific queries. It will discard repetition to reduce the volume of information while retaining important content. This engine requires the interaction of human for final decision making.

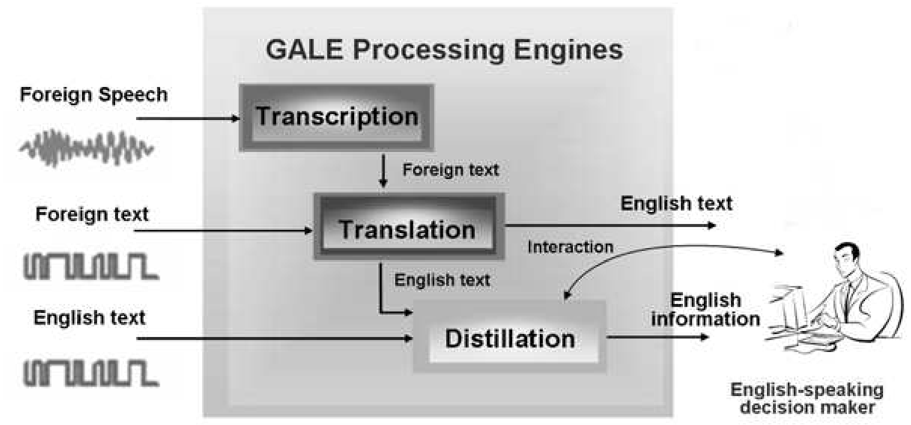


FIG. 3.2: Gale Processing Engines

Distillation is a concept entirely new to GALE, in which relevant information is extracted from foreign language and English input and concisely presented to the user in English. GALE distillation is not a key-word search, and does not involve summarization. Instead, it utilizes language analysis techniques to identify information relevant to a user's query, with the aim of extracting all available relevant information and nothing redundant. GALE's distillation engine will allow computerized searches that will return just the specific context and sense desired, even when the text was not originally in English. For example, an analyst will be able to ask the GALE distillation engine to produce a biography of a person, provide information about an organization during a specified period of time or to find statements made by certain people during specified times on designated topics.

There are some organizations which support the background information of this project. The Linguistic Data Consortium (LDC) is an open consortium of universities, companies and government research laboratories who create training corpora, guidelines, annotation tools and related resources to support the Distillation task. Training resources include:

- Collections of raw source text that serve as input to training and test corpora for Distillation
- Queries that conform to designated query templates
- Manual annotation in English, Chinese and Arabic for:
  Relevant documents
  Snippets of relevant text
  Formalized phrases that express core facts extracted from those snippets
  Logical deductions and relationship extraction between pairs of these phrases (The presence of one implies the second one to be true)
- Relevance judgments of system-extracted snippets

In GALE project, the ultimate performance targets are to translate Arabic and Chinese speech and text with 95% accuracy and an extremely high degree of consistency (90-95%), and to extract and deliver key information with proficiency matching or exceeding that of humans. GALE systems must be able to perform at these high levels of accuracy and consistency for foreign language information from a wide range of domains and genres, and be able to cope with informal part of language.

3.3. Document Triage

Document triage in foreign language is one of the biggest existing problems, and the GALE approach attacks the problem in the reverse order. In a distillation process, it seems appropriate to search and triage first and translate only after the important documents have been identified.

Basis Technology provides foreign-language conversion for intelligence that focuses on name translation. Because of the many nuances and broad differences between languages, implicit meaning can be ambiguous and translations are not always accurate. In addition, the English spelling of foreign words varies from system to system and the engine should take care of these subtle points. There is also a difference between orthographic accuracy and phonetic accuracy in transliteration. How to pronounce a word is different from how to write or how to translate it. So analysts can risk missing important information because of the potential for inconsistent transliteration. Additionally, in some languages such as Arabic, there is no upper case, lower case or hyphen. Rather, these are impositions by the English translation as a way to make the name more understandable to English speakers.

While the research and development of language translation technology is an ongoing effort, there is no question about the need for it.

DARPA's GALE program is directly addressing a cross-agency requirement for increased foreign language skills and searchable language-translation technology.

3.4. Template-based Queries

GALE distillation queries pertain to GNG(Go/No-Go) distillation engine evaluation. Each query uses a query template among 17 predefined templates.

For each template, there are some categories that indicate the reason for information relevance. To attribute a sentence as relevant to a query, at least one category must apply. Training data will not include category annotations and GALE distillation engines are not required to provide category labels in their output.

Given a query and a particular sentence, the task is to classify the sentence in one of the two classes of relevant or irrelevant.

ICSI and SRI manage to work on queries of templates 1, 8, 12, 15 and 16, however the focus of this work is on template 1. In the following, a number of query templates are listed:

For example, in Template 1 queries, *event* is the variable portion of the query. Relevant information focus on the event, the persons, places and activities associated with this event. Here the goal of a distillation system is to output ordered segments called snippets that can be considered as an answer to the query. A snippet can range from a fragment of a sentence to a paragraph. Below is an example query (in which the event is variables) with some related snippets:

```
LIST FACTS ABOUT [event](template 1)
WHAT [people|organizations|countries] ARE INVOLVED IN [event]
AND WHAT ARE THEIR ROLES?
PROVIDE INFORMATION ON [person/organization]
FIND STATEMENTS MADE BY OR ATTRIBUTED TO [person] ON [topic(s)]
DESCRIBE THE RELATIONSHIP OF [person/org] TO [person/organization]
DESCRIBE INVOLVEMENT OF [person/organization/country] IN [event/topic]
DESCRIBE THE PROSECUTION OF [person] FOR [crime](template 8)
HOW DID [country] REACT TO [event]?
DESCRIBE THE ACTIONS OF [person/organization] DURING [date] TO [date]
PRODUCE A BIOGRAPHY OF [person] (template 12)
DESCRIBE ARRESTS OF PERSONS FROM [organization] AND GIVE THEIR ROLE
IN THE ORGANIZATION (template 15)
DESCRIBE ATTACKS in [location] GIVING LOCATION, DATE, AND NUMBER OF
DEAD AND INJURED (template 16)
WHERE HAS [person] BEEN AND WHEN?
```

FIG. 3.3: Examples of query templates. The bracketed words are the variable parts.

Query:

- LIST FACTS ABOUT EVENTS DESCRIBED AS FOLLOWS: [the dual suicide bombing at a concert in Moscow] formation since [28 Sept 2000].

Snippets:

- At least 17 people were killed and 22 others wounded in suicide bombings.
- In Russia's capital of Moscow on Saturday Police put minimum power of each of the two bombs at equivalent of 500 grams of TNT.
- The Russian government blamed the attacks on Chechen rebels.

One critical component for distillation is detecting sentences to be extracted from each relevant document. The user typically is not interested in reading the whole news story but instead just the portion with the requested information content. This process is relevant but not exactly equivalent to document summarization, information retrieval, question answering, or information extraction. In a summarization system, there is usually no predefined query; simply, documents are summarized. Yet, sentence extraction may be part of a summarization system. However, the goal in a summarization system is to extract the most informative sentences that summarize the whole document, so features like the number of named entities in the sentence or rank of the sentence in the document are extremely useful. Recently question-focused (or query relevant) summarization in also incorporated in the evaluations where the similarity of each sentence with the question can also be used as a feature. However, the questions do not have a predefined structure such as the query templates. Question answering systems, on the other hand, act upon a requested question, such as list all the European union countries. In that sense question answering output is very formatted compared to distillation. Distillation usually requires details for the event in question. Even for a query very close to question answering, such as What is the relationship of Sarkozy to France, the answer is not a single word or sentence (such as He is the president); instead, anything relating Sarkozy to France in the corpora needs to be in the answer set. Information extraction, more specifically event extraction, can be considered as the closest match for distillation. Some of the ACE events (such as arrest or attack) are among the GALE distillation queries.

3.5. Related Work To Distillation

A large body of work in question-answering has followed among which a number of systems have used techniques inspired by information extraction. Below there is the description of some prior work which help in improvement of distillation techniques eventually.

3.5.1 Selecting On-Topic Sentences from Natural Language Corpora

One approach is a system that answers the *on-topic* questions for natural language documents and arbitrary free-text topic descriptions. The approach is based on creating proposition trees, semantically organized hierarchies of predicate-argument structures, for topic formulation and document sentences.

The biggest advantage of this system is for the question answering task and for the kinds of queries that either are focused entirely on a topic or event such as, "List facts about [EVENT]" or those who inquire about a particular aspect of a topic or the reaction an event for ex.: "How did [COUNTRY] react to [EVENT]". In the example queries above, [EVENT] is a slot filled with a free-text description of some event. To select documents from a corpus that answer such queries, it is required to say how "on topic" they are with respect to such free-text formulations.

Matching a slot can be trivial if the slot consists of a single name or very specific term, but it can also require considerable effort when the slot contains a complex description of some topic or event. An example of a difficult case is the following query from the GALE Distillation task: List facts about [the looting of Iraqi museums after the U.S. invasion]. Another difficulty in dealing with queries that have topic and event slots, is the fact that human language makes it possible to convey on-topic information without using the words that were used in the actual topic formulation. Here is an example of a sentence related to the query slot [the looting of Iraqi museums after the U.S. invasion]: To resolve this problem, free-text query slots can be represented

```
Many works of art were stolen from Baghdad galleries in 2003.
```

as predicate-argument *proposition trees*. Responses to queries involving free-text arguments can be found using a matching algorithm to instantiate the query proposition tree in the proposition trees of the candidate responses.

Predicate-argument structures (propositions) can be extracted from syntactic parsers. Each proposition has a type (e.g. VERB, NOUN, MODIFIER, SET and others), a head predicate and a non-zero number of arguments. Each argument refers to another proposition or a mention (associated with a subtree of the sentence's syntactic parse). Each argument also has a role, where the lexicon of roles consists of a closed set of grammatical functions such as subject, object, indirect object and others.

Individual propositions extracted from a sentence can be assembled into one coherent tree structure. This tree represents the semantic interpretation of the sentence, abstracted from its exact syntactic realization. In practice, a sentence may consist of either one or several such proposition trees also known as proposition forest, created by replacing mention arguments of propositions with propositions referencing those mentions as their predicate head. In order to examine the degree of topicality, the proposition forests are extracted for both slot and sentence. then each of slot proposition trees τ_i in each of the sentence proposition trees t_j . If an instantiation is possible, its score θ_{ij} is compared with other instantiation scores for this slot proposition tree, and the non-negative maximum is saved:

$$\theta_i := \max(\max_j \theta_{ij}, 0). \tag{3.1}$$

The topicality measure is defined by the following formula:

$$\theta := \frac{\sum_i w_i * (\theta_i / \theta_i^{max})}{\sum_i w_i}. \tag{3.2}$$

Where w_i is the weight of the proposition tree τ_i. This weight depends on the number of nodes. In this work, two more topicality measures are also introduced where each complete others in terms of precision/recall characteristics.

Topicality measure η is based on subtrees of full proposition trees and topicality measure μ indicates the percentage of tree nodes in a slot forest whose word pool has at least one match with the word pool of sentence forest. A sentence is said to be relevant on a given topic if one of three measures exceeds a certain threshold(Levit *et al.*, 2007b).

3.5.2 Exploiting IE Annotations for Document Retrieval in Distillation Tasks

This approach uses information extraction annotations to augment document retrieval for distillation. The fact that some of the distillation queries can be associated with annotation elements introduced for the NIST Automatic Content Extraction (ACE) task, is the base of this approach.

The objective of the ACE is to develop technology, to automatically infer from human language data the *entities* being mentioned, the *relations* among these entities that are directly expressed, and the *events* in which these entities participate.(Doddington *et al.*, 2004). *Entities* include person, location, organization, and other names.*Value* is a text string that further characterizes the properties of some Entity or Event.*Mention extraction* aims to group entities using nominal, pronominal, and named representations.*Relation extraction* tries to find predefined relations between the mentions such as wife of a person or employee of an organization. *Event types* include: Life, Movement, Transaction, Business, Conflict, Contact, Personnel, Justice.

The data includes both textual and audio data in multiple languages, namely English, Chinese, and Arabic. Non-English data, is being translated automatically. The University of Massachusetts INDRI search engine indexes all the data. During runtime when a query is given, the INDRI search engine retrieves candidate documents, considering the dates, the sources of documents to be searched, and so on, as specified in the query. When the relevant documents related to query are returned, the goal of sentence extraction is to tag each sentence in these documents with respect to its relevance.

Here the statistical method for sentence extraction in information distillation is data-driven and uses lexical and simple semantic features and treats the problem as a binary classification task, where each sentence is classified as relevant (positive) or not relevant (negative). To train the sentence extraction models, the negative and positive examples from the given answer keys are extracted, which have the relevant snippets and the corresponding document identifiers for each query. As the relevant sentences in those answer keys also include the document identifiers, all sentences in those documents are extracted.

The important change is introducing an intermediate processing stage between the INDRI information retrieval engine and the sentence extraction module, to filter out irrelevant documents. exclusion of irrelevant documents at this stage, does not hurt neither precision nor recall(Hakkani-Tür *et al.*, 2007).

3.5.3 Statistical Sentence Extraction For Information Distillation

One of the approaches to distillation is a statistical sentence extraction. Basically, it can be considered as a classification problem, where each candidate sentence in documents is classified as relevant or irrelevant to a specified query. The documents may be in textual or audio format also in other languages rather than English (such as Chinese and Arabic). For audio documents, both manual and automatic transcriptions are used. For non-English documents, automatic translations is the prior step before classification. The University of Massachusetts INDRI search engine indexes all the data (Strohman *et al.*, n.d.). During runtime, with a given query, the INDRI search engine retrieves *candidate documents* considering the dates, the sources of documents to be searched, and other relevant categories specified in the query template. Then the sentence extraction process tries to identify *the potential snippets*. Finally, similar sentences are clustered into groups. Due to the diversity of the data sources and the noise introduced via speech recognition (ASR) and machine translation (MT), it's essential to have a robust method.

In one of the approaches which is sentence extraction by keyword-spotting, each sentence gets a vote if certain keywords and named entities, depending on the template, are mentioned in the query appear in the sentence. Then the recall/precision curves are drawn according to the number of votes. However this approach requires human involvement for in-domain knowledge. In the second approach which is based on classification, a data-driven (or learning) method for sentence extraction is employed in information distillation. The negative and positive examples are extracted from the given answer keys which have the relevant snippets and the corresponding document identifiers for each query. Since the relevant sentences in answer keys also include the document identifiers, all sentences in those documents are extracted as examples, and the sentences whose portion are in the answer key are marked as positive examples, and all the rest as negative examples.

During classification, lexical and semantic features are used. Lexical features consist of word n-grams obtained from the training examples. Using all word -grams instead of several keywords as features is also expected to improve the robustness of the system. This can be considered as a query-specific information extraction system. Then these features are augmented with semantic ones by tagging the raw sentences to mark the instances of organization, location, or dates in the query.

The classifying tool of this approach is *AdaBoost* which performs a discriminative classification method with both lexical(word n-grams) and semantic features. Boosting is an iterative procedure, on each iteration a weak classifier is trained on a weighted training set, and at the end, the weak classifiers are combined into a single, combined classifier.

One problem with this task is that it is sometimes not clear whether a sentence must be extracted since it is a subjective decision. So instead of making a binary decision, the data may be divided into more classes such as very relevant, marginally relevant, and so on. However, the results of classification based approach for information distillation, indicate 11%-13% relative improvement over a baseline keyword-spotting-based approach(Hakkani-Tür & Tur, 2007).

3.5.4 Q.A. using Integrated Information Retrieval and Information Extraction

This research provides extended responses to questions regarding specialized topics. This task is a combination of information retrieval, topical summarization, and information extraction.

This technique interleaves information retrieval (IR) and response generation, at the first stage using IR in high precision mode to return a small number of documents that are highly likely to be relevant and can determine what name variations are used in the corpus and estimate how many documents contain relevant references.

Information extraction of entities and events within these documents is then used to point highly relevant sentences and associated words are selected to revise the query for a second pass of retrieval, improving recall. The relevant context is approximated by measuring the proximity of the target name in the query and extracted events.

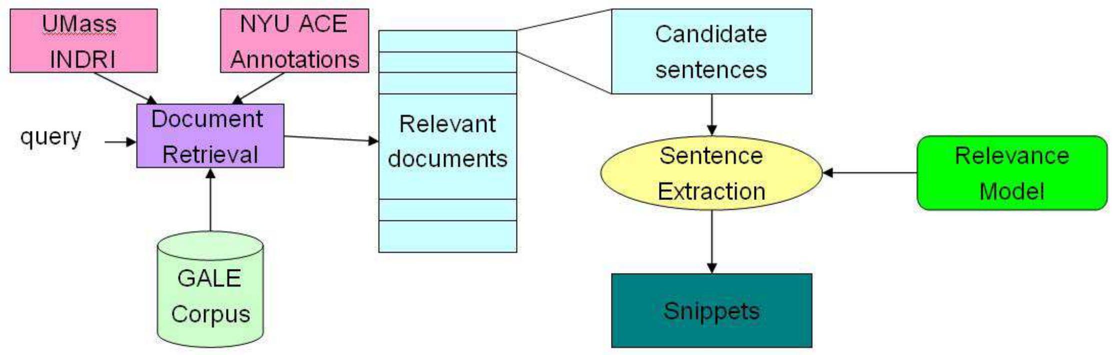


FIG. 3.4: IXIR Approach.

Once the document retrieval is completed, the information extraction component produces the full range of annotations as specified for the ACE 2005 evaluation, including entities, values, time expressions, relations, and events. The extraction engine identifies seven semantic classes of entities. Each entity will have one or more mentions in the document; these mentions include names, nouns and noun phrases, and pronouns. Text processing begins with an HMM-based named entity tagger, which identifies and classifies the names in the document. Finally, the event classifier (which uses the proposed arguments as features) is used to reject unlikely event sequences.

Once the final set of documents is obtained, the answer generator module selects candidate passages. The names, with alternate renderings, are located through the entity mentions by the IE system(Schiffman & McKeown, 2007).

3.5.5 Integrating Several Annotation Layers For Statistical Information Distillation

In this research a sentence extraction algorithm for Information Distillation a presented. For a given template based query, relevant passages must be extracted from massive audio and textual document sources. For each sentence of the relevant documents, the statistical classification methods are used to estimate the extent of its relevance to the query. Two aspects in relevance are considered: the template (type) of the query and its slots (free-text descriptions of names, organizations, topic, events and so on, around which templates are centered). The important aspect of this work is the choice of features used for classification. The features come from a combination of sentence annotations from several levels (such as word transcriptions, syntactic and semantic parses, and IE elements) into time-synchronous directed acyclic graphs (charts) to learn sentence relevance. By using the system for GALE Distillation, the average F-measure for five templates are improved from 0.39 (with words only) to 0.51 which is around 30% relative progress(Levit *et al.*, 2007a).

This approach is called IXIR. Figure 3.4 shows how this approach works.

CHAPTER
4

Approach

This chapter consists of presentation of methods to improve the performance of information distillation system. In this chapter, first we have a quick review on the distillation system and query types that are the target of this work. Then alternative solutions proposed during the project will be discussed.

4.1. The Baseline Supervised Approach for Template 1

This work is mostly dedicated to improve the performance of answering to *Template 1* queries of GALE project. Template 1 queries look for responses to "Describe the facts about [EVENT]", with possible definitions of EVENT slot such as "Stolen art works from Baghdad galleries in 2003". The output of the distillation system is a list of snippets (sentences) relevant to that query.

Template 1 queries are hard to answer with the current IXIR distillation approach based on supervised classification. Contrary to other templates, its form is very general. Answering to these queries is more similar to passage retrieval of an IR system. However, if the performance of classification for this template is increased, it would be useful to answer free-template questions as well.

In the IXIR approach, to perform the classification task, one needs to train the classifier on data that have been already labeled. Usually,the amount of data used to train an efficient classifier for sentence classification is large; at the same time, labeling data is a very time consuming and labor intensive task that has to be done by humans. Implementing an efficient unsupervised classification method to annotate data automatically, reduces the labeling effort for more sophisticated approaches. Other templates may also exploit similar unsupervised methods.

In every classification method, once a learning algorithm has been selected, the next step is to feed this algorithm with a set of training data. The data set is split into instances (also called examples) which are represented by a set of features and a class. For the sentence classification task, the features are mainly word N-grams and synonyms.

However, for free form questions such as Template 1, it also is difficult to design features that can capture the characteristics of a response. There is no guarantee that the relevant responses to a new question will have the same characteristics as the relevant responses for other questions. For example, for Template 16, which deals with attacks in a specific location, attacks in *London* and *Moscow* both contain the number of people died or wounded in the attack among the relevant responses. However, such patterns are hard to form for Template 1 questions, which may request a description of *September 11* or *election of Hillary Clinton to the U.S. Senate*.

To answer the query Q_i by a supervised method, it is required to use the answer keys provided for $\{\bigcup Q_j, j \neq i\}$. If the classifier uses only the word n-grams as features, due to the large number of words in word pool, it may not necessarily learn the words particularly related to Q_i. In other words, the training method is not focused on important features for Q_i.

To improve the search quality, other sophisticated approaches are also proposed. Instead of just using word transcriptions and names, they use some additional features like syntactic parses, semantic predicate-argument structures, and various elements of Information Extraction (IE) annotations (Levit *et al.*, 2007a). Although these features have augmented the performance of classifier, the same problem still exists where the query Q_i is not targeted specifically. Therefore, it's not assured to be answered with appropriate words transcriptions and features.

In this study we employed AdaBoost which is a linear classifier. A linear classifier, classifies to group items that have similar feature values and achieves this by making a classification decision based on the value of the linear combination of the features. Each instance of query is a vector of the feature space. For a given experiment, all instances must be described in the same feature space. After T iterations, boosting outputs a function $f(x)$ which computes the probability $p(C_r|x)$ that the instance x is hypothesized as a sentence boundary.

An *acceptance threshold* t_{acc} is then used to map this probability on either a relevant or irrelevant class, and thus decide of the class $C(x)$ of the instance x:

$$C(x) = \begin{cases} C_r & \text{if } p(C_r|x) > t_{acc} \\ C_{ir} & \text{otherwise} \end{cases} \tag{4.1}$$

where C_r stands for the relevant sentence to be labeled as class 1 and C_{ir} for irrelevant sentence to be labeled as class 0.

The acceptance threshold t_{acc} is a parameter that can be optimized on the training or held-out set. If it is set on held-out set, a lower acceptance threshold leads the final hypothesis to contain more data in training set, while a higher acceptance threshold on the contrary leads to less training data. It can also be used to give a precise profile to the classifier by balancing the performance between the recall and the precision, two metrics that are used very frequently in classification evaluations.

In the case of supervised classification, to be able to learn how to classify the instances according to their features, the algorithm needs to be provided with the positive and negative classes of the training instances. i.e. with instances that have been labeled. Figure 4.2 shows the steps of supervised learning concretely:

4.2. Unsupervised Learning for Distillation

In unsupervised classification, supposedly positive and negative labels should be generated by the algorithm itself. The *data files* required for training, consist of several documents, each of them containing some sentences which usually come all together from a news feed. For template 1 queries, by definition, annotated training data are not accessible. Therefore, we need to form the query-specific training set with estimated labels, as many as possible, by various techniques which will be explained in this chapter. Training set will be expanded and developed more precisely during several iterations. We call this approach *incremental classification*. In this type of work, a kind of blind feedback is generated to reply to query, and the performance of classifier highly depends on the accuracy this feedback. Although we already have blind feedback for

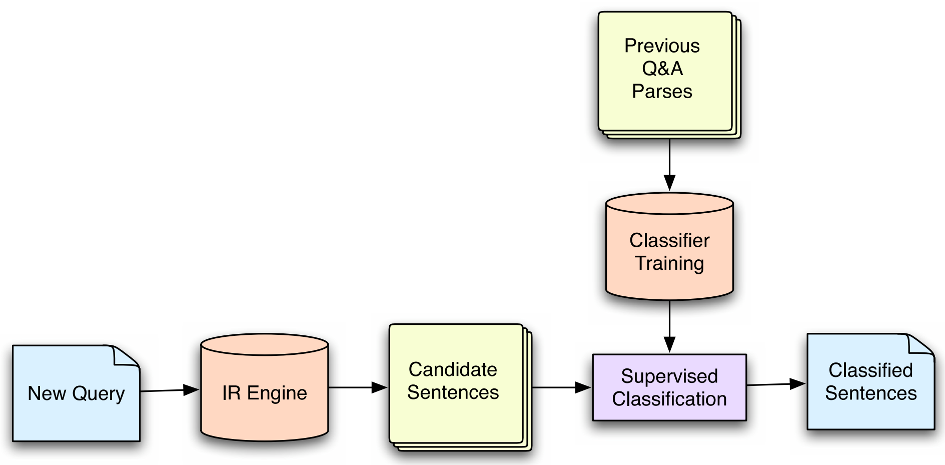


FIG. 4.1: This diagram shows how supervised methods perform the classification.

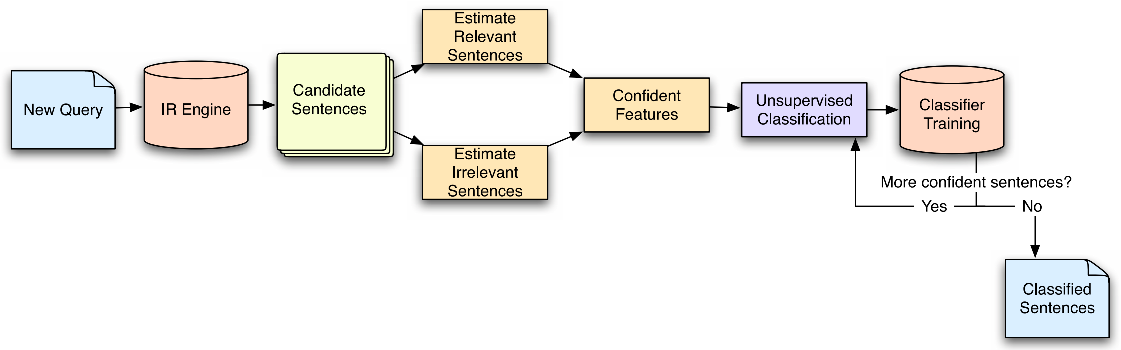


FIG. 4.2: This diagram shows how our unsupervised classifier works.

document retrieval, applying it to information distillation via developing a statistical classifier is a novel approach.

To understand how the proposed unsupervised learning method works we try to explain it by an example. Assume that we are given a database of documents with the set of sentences S in each document and a new query slot of desired template. The goal is finding relevant sentences for that query slot. We start with a small set of rules to find out the sentences that have the highest likelihood of being an answer to this query. As a preliminary set of relevant sentences, we may consider the ones that contain the exact phrase in the query slot. We call this the set A. Then we can also find a set of sentences that have a very high likelihood of being irrelevant to the query, such as the ones that do not have any overlap in terms of words with the words in the query slot, after excluding the stop words. We call this set B. At this step we label the sentences of set A as 1, and those of B with 0 class. The next step would be training a classifier with these highly confident examples, estimating labels to the examples in S or the examples in $S - (A + B)$ iteratively via trained data obtained by the classifier. The goal here, is to iteratively refine the classification decisions.

For a given query, Q, when we talk about data file, *Q.data* file, which contains candidate documents for query Q returned by either LDC or University of Massachusetts INDRI IR engine. Documents returned by LDC are known to be relevant however in the case of INDRI, information about relevance is not available completely. Though, the case of INDRI should be closer to reality.

For all the proposed unsupervised learning methods, before training the classifier we need to clean the query texts and data files to prevent noise effect in classification as much as possible. For example, in words such as "can’t", "Clinton’s", to prevent losing the important words, it is better to remove the " ’ " for methods using TF-IDF.

In all parts of this project, frequencies of word occurrence have a very important role. When we talk about words, we mean important words to a particular query. For instance in a query like "The looting of Iraqi Museums following the 2003 US invasion" , the words that are worth to be looked for, are "looting", "Iraqi", "Museums", "US" and "invasion".

In the above example, we have removed stop words and numbers. Although "2003" may be an important word by itself, due to the structure of test files, we have to eliminate it. The reason is that the date of an event is usually reported in document header. Including this date causes lots of irrelevant sentences annotated as 1 in initial approximation. Therefore a considerable augmentation of error rate will follow.

To extract the non-function words automatically, we use a tool called *WordNet*. More explanations about WordNet are provided in the rest of this chapter. As we apply WordNet on a text, stop words and numbers are automatically removed, since WordNet just recognizes the four structures of verb, noun, adjective and adverb. Also for some examples such as "Iraqi", WordNet may return several stems for different grammatical roles such as "Iraq", "Iraqi", etc. In the methods that require stems, we usually choose the one with the shortest length to be able to have wider search results from our text. For the rest of cleaning process, we use regular expressions. As an example, the above query transforms into "loot iraq museum follow us invasion".

We have tried several methods for the following steps. For example, in a few of methods, next step is to compute the occurrence frequency of each remaining word in the corresponding data file. For each query we take the processed query words and compute the frequencies. Then we take their mean. Among the words in word pool, we extract those who have a frequency higher than mean value.

In this study, to optimize error rate and F-measure we use five types of thresholds:

- Mean Threshold: This parameter leaves a margin below mean base level of frequency of word occurrences, and lets choose words with frequencies bigger the margined mean.
- TF-IDF Threshold: Same as Mean Threshold, just leaves a margin below TF-IDF base level.
- Number of iterations of classifier.
- Number of iterations of algorithm
- Probability threshold: This parameter which is also called acceptance threshold. th_p defines the probability above which, estimated classified sentences can be judged with certainty as true classified ones.

Depending on the method, number of selected thresholds may vary.

For methods using TF-IDF, instead of using mean frequency, important words are chosen by a threshold on TF-IDF. Further, for methods that use TF-IDF, other than query stemming, we find document stems to obtain even cleaner data and more exact TF-IDF values.

Up to now, methods use a word selection more strict at the beginning and they they remove the constraints gradually. It means that they start word selection by an upper limit, they they reduce(relax) the number of words they are looking for. Contrary to this type of word selection, in our last method, which will be described more in detail, the words are selected from at least 1 word and then their number increases gradually in different iterations.

WordNet WordNet is a semantic lexicon for the English language. It groups nouns, verbs, adjectives and adverbs into sets of synonyms called synsets, each expressing a distinct concept.

It provides short, general definitions, and records the various semantic relations between these synonym sets.

Synsets are interlinked by means of conceptual-semantic and lexical relations. Every synset contains a group of synonymous words or collocations where a collocation is a sequence of words that go together to form a specific meaning. Different senses of a word are in different synsets. The meaning of the synsets is further clarified with short defining glosses (Definitions and/or example sentences). WordNet distinguishes between nouns, verbs, adjectives and adverbs because they follow different grammatical rules. Most synsets are connected to other synsets via a number of semantic relations.(Fellbaum, 1998)

The purpose of WordNet is to produce a combination of dictionary and thesaurus that is more intuitively usable, and to support automatic text analysis and artificial intelligence applications. Although WordNet contains a sufficient wide range of common words, it may not cover special domain vocabulary.

In this work, we use this tool for two different purposes. The first one as explained, is to find the stems of words for a clean search. The more important one is the collection of synonyms that it returns. Later, we will use this collection as a semantic feature for our classifier.

Back to word selection, once the important words are selected and processed, we start generative (incremental) classifier by this set of high confidence examples. Our incremental classifier starts with an initial annotated file. This file is obtained by different methods described in the proceeding methods. The incrementing process is done during different iterations, but the general concept is quite similar for all the methods. We explain it just once for the case of "M1: Strict Initial Classification For Relevant and Irrelevant Snippets" method presented below.

4.2.1 M1: Strict Initial Classification For Relevant Snippets

In this method, we extract the sentences of original data file (F_0) which contain *all* the non-function words in the query and label them as relevant (class 1). We label the rest of the sentences as irrelevant (class 0). The union of these labeled snippets, form the initial training file for classification (F_1).

We then train a bootstrap classifier using this initial data. Later, we automatically classify the whole corpus omitting the set of previously used sentences. Each sentence in the corpus is weighted either to be relevant (positive) or to be irrelevant (negative) by a coefficient calculated by classifier. These posterior probabilities are calculated as in the following formula:

$$p(l|x) = \frac{1}{1 + e^{-2w.iter}} \tag{4.2}$$

where *iter* is the number of iterations used to train the classifier and w is the final classifier output by the learning algorithm.

Next step is to select a set of confident sentences in both relevant and irrelevant classes. This is a crucial point in our algorithm. We add-up the sentences which are confidently classified to the existing training data. We optimize the confidence threshold using a held-out set. Then we retrain the classifier using this extended set. This makes the classifier to *learn new words* other than those already found. Then the same process is applied in an iterative fashion. In our experiments we limited the number of iterations to 3, since no further learning happens after that.

The pseudo code of this operation is shown in algorithm Algorithm 1.

Algorithm 1 Auto Incremental Classifier, Method M1

```
Given the query and related data : ($Q$ , $Q.data$)
FindQueryWordFrequency($Q$ , $Q.data$)
$new\_mean$ = $meanWordFrequencies$) $\times$ $(1 - meanThreshold)$

for each $w_i = 1...w_n$ do
    if $freq(w_i) \geqslant new\_mean$ then Add $w_i$ to WordPool
    end if
end for

FirstIteration

$Sen1 = \emptyset$
$Sen0 = \emptyset$
if $\forall w_i \in$ WordPool exists in $sentence_i$ then Label $sentence_i$ by 1; $Sen1 = Sen1 \cup sentence_i$
elseLabel $sentence_i$ by 0; $Sen0 = Sen0 \cup sentence_i$
end if

Label the rest of the sentences as 0
$dataFile_1 = Sen0 \cup Sen1$
Remove all $w_i$ in WordPool from $dataFile_1$.

Train the classification model using $dataFile_1$
Classify the rest of the sentences

NextIterations

Use Output of Classifier at Previous Iteration
if $prob(sentence_i) \geq (1 - prbThreshold)$ then $dataFile_{i+1} = dataFile_i \cup sentence_i$
end if
if $prob(sentence_i) \leq (0 + prbThreshold)$ then $dataFile_{i+1} = dataFile_i \cup sentence_i$
end if
Train the classification model using $dataFile_(i + 1)$
Classify the rest of the sentences

Repeat the Iteration
```

4.2.2 M2: Strict Initial Classification For Relevant and Irrelevant Snippets

This method is more strict comparing to the previous one in terms of labeling initial irrelevant snippets. Once we find the collection of relevant sentences, instead of labeling all the rest as irrelevant, we extract the sentences that have *none* of the non-function words and label them as irrelevant. By this method, the initial data file is smaller, but it is less noisy and we expect to obtain better performance.

The portion of pseudo code which is different from previous method is shown at Algorithm 2

Algorithm 2 Auto Incremental Classifier, Method M2

```
Given the query and related data : ($Q$ , $Q.data$)
......

$Sen1 = \emptyset$
if $\forall w_i \in$ WordPool exists in $sentence_i$ then Label $sentence_i$ by 1; $Sen1 = Sen1 \cup sentence_i$
end if

$Sen0 = \emptyset$
if $\nexists w_i \in$ WordPool exists in $sentence_i$ then Label $sentence_i$ by 0; $Sen0 = Sen0 \cup sentence_i$
end if
.....
```

4.2.3 M3: Relaxation of Initial Classification For Relevant and Irrelevant Snippets

This experiment is a bit different from previous ones. It starts like M2 for the first iteration, but in second and later iterations we change the strategy to enlarge the training data file. Instead of using the output of classifier, we try to relax the search criteria and recreate a larger training set. For example, if in the first iteration we look for the presence of 4 words to mark a sentence as relevant, in second iteration we look for the presence of the 3 highest frequency words, then for 2 highest frequency ones, etc. We continue for three iterations or we stop when we have at least one word to look for before third iteration. The difference of this method with previous ones is removal of two threshold; number of algorithm iterations and probability threshold. However, mean threshold and classification iterations are still applied.

4.2.4 M4: Word Selection By TF-IDF At Document Level

The TF-IDF weight (Term Trequency/Inverse Document Frequency) is a weight often used in Information Retrieval. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is limited by the frequency of the word in the corpus. The Term Frequency in the given document is the number of times that term appears in the document. This count should be normalized. Normalization is to prevent a bias towards longer documents which may have a higher term frequency regardless of the actual importance of that term in the document. The equations to calculated TF and IDF are mentioned in 4.3 and 4.4.

$$TF_i = \frac{n_i}{\sum_k n_k} \tag{4.3}$$

where n_i is the number of occurrences of the considered term, and the denominator is the number of occurrences of all terms.

$$IDF_i = \log \frac{|D|}{|\{d : t_i \in d\}|} \tag{4.4}$$

where $|D|$ is the total number of the documents and $|d_i|$ is the number of documents in which term t_i occurs at least once.

Considering the file $Q.data$ containing some documents, we find the $TF - IDF$ for each $t_i \in Q$. Term frequency is limited by number of documents. This makes sure of reducing the weight of frequently repeated words such as stop words. In this method, we select the words with TF-IDF larger than predefined threshold. Once the important words are extracted, we look for word N-grams up to $N = 2$ in the document to find the relevant sentences and for then initial data file. Once this part is done, the rest of the method is similar to the previous ones. Again, we use the output of classifier to increment the size of training documents and reiterate the classification.

4.2.5 M5: Word Selection By TF-IDF At Sentence Level

This method's concept is very similar to the previous one, except that we replace documents with individual sentences. TF-IDF produces more accurate results when number of documents is high enough. Since the size of our corpus is not that large, we tried to simulate having a high number of documents by this approach. Once the TF-IDF in sentence level are calculated, we extract word N-grams up to $N = 2$ and continue as method M4.

4.2.6 M6: All-Words Selection

This method is very similar to M2 except that there are some changes for word selection procedure. The significant change of this method is including semantic features. We include not only word transcriptions but also word synonyms found by WordNet. For each query word, we obtain up to two synonyms (the most pertinent ones) from WordNet and stem them. Then we look in text just for synonyms and keep those that are repeated sufficiently. For computing a measure of sufficiency, we do the same as method M2 does for main words. We find the occurrence frequencies of synonyms in the text, compute the mean of these frequencies and select the synonyms that occur more than this mean in the text. It should be noted that synonyms are repeated much less frequently than basic words and shouldn't be compared to the number of occurrences of basic words. Once we form a complete collection of words, we look for word unigrams and bigrams in the text. In fact, by using bigrams in addition to unigrams, we relax the classification. We find relevant sentences to union of unigrams and bigrams and add them up. Then we remove the redundant sentences. Since the relaxing is done at word count level, we also dropped the mean threshold. The pseudo code of this method is presented below:

1. For each word w_i in word pool, find the first two senses of synonyms.
2. Find the frequencies of synonym words in document collection.
3. Take mean of them.
4. Add those synonyms who occur more than their frequency to word pool and name this set by W.
5. Look for sentence that contain a combinations of w_1w_2, w_2w_3, ... , $w_{n-1}w_n$ PLUS $w_1w_2...wn$ in text.
6. Remove redundant sentences and label the rest as 1.
7. Look for sentence that do not contain $w_1w_2...wn$ in text.
8. Label them by 0.
9. Add up these two set of sentences and create F_1.
10. Continue the classification as previous methods.

4.2.7 M7: Loose Word Selection

In all of the previous methods, although we tried several alternatives, there were some cases that the classifier was not able to detect relevant sentences for the initial data file. It happens in cases that keywords passed through thresholds are not present in text.

This weakness causes a lack of learning relevant sentences in the first iteration, consequently classifier never learns correct features even in next runs. In mathematical terms, it leads to a zero value for precision and recall and eventually zero F-measure. Even if we obtain good results for F-measure in some of test queries, these zero values have a negative impact on the average F-measure of test queries calculated later in n-fold. Therefore, we decided to try a method to have at least one true positive sentence. For this, it is required to relax the search criteria since the beginning. Contrary to previous methods that we started with a large collection of words and relaxed afterwards, this time we start with the least number of words and make a more strict search at later runs. For example at first iteration, we look for 10% of the query words or at least one word to appear in a sentence and label such sentences as relevant. Then we label the sentences that do not carry any of the query slot words (after stop word removal and stemming) as irrelevant. We train an initial classifier, and proceed with the iterative step. Here, the percentage of words is a parameter and is optimized according to the development set.

CHAPTER
5

Data and Metrics

This chapter describes qualitatively and quantitatively the corpus that we use for the classification. Then we proceed with feature types related to this work, as well as explanations of some important measures. Then we explain the performance factors that we use to measure the performance of unsupervised classification. Performance factors are compared to the baseline which will be described afterwards. Finally, we explain about a statistical significance test which will be used later in results chapter.

5.1. Data

All the experiments of this work are done in two phases. For the first phase, the corpora are a combination of TDT4 and TDT5 corpora from LDC (we will refer to them as Y1-data) along with a few more queries from the corpus of GALE-Y2. This corpus includes the collection of English written news forms including blogs. (Wayne, June 1998). For each query we only consider those documents in which, LDC found at least one relevant sentence. During the initial experiments, we use the documents which come from Y1 data source as both training and test data sets. For this part, we classify the sentences related to 27 queries of template1. Later, we add to our corpus 10 queries from GALE-Y2. Once these queries are added to our corpus, they form our test set and in the meanwhile the previous 27 queries form the training set. The important point is that the queries in test set should not be included in training set.

In the second phase, experiments are repeated on documents returned by UMass INDRI Document Retrieval engine. The latter set has the advantage of being conducted under more reality-like conditions.

5.1.1 LDC Documents

Linguistic Data Consortium (LDC) has annotated part of their TDT4, TDT5 and GALE-Y2 corpora with respect to a number of queries. For each query, it employed an off-the-shelf search engine to find documents with at least one relevant sentence in it. Then all sentences of these documents were annotated with respect to the query. In other words, given a query of $template_i$ as input, it produces a number of binary "yes/no" decisions for each sentence of the retrieved documents. *Template* of the query and its *slots* were both taken into account while making relevance decisions.

To have a general idea about the documents returned by LDC, we have depicted the proportion of relevant and irrelevant sentences per query in 5.1. As can be observed, for each query, at least some relevant sentences exist.

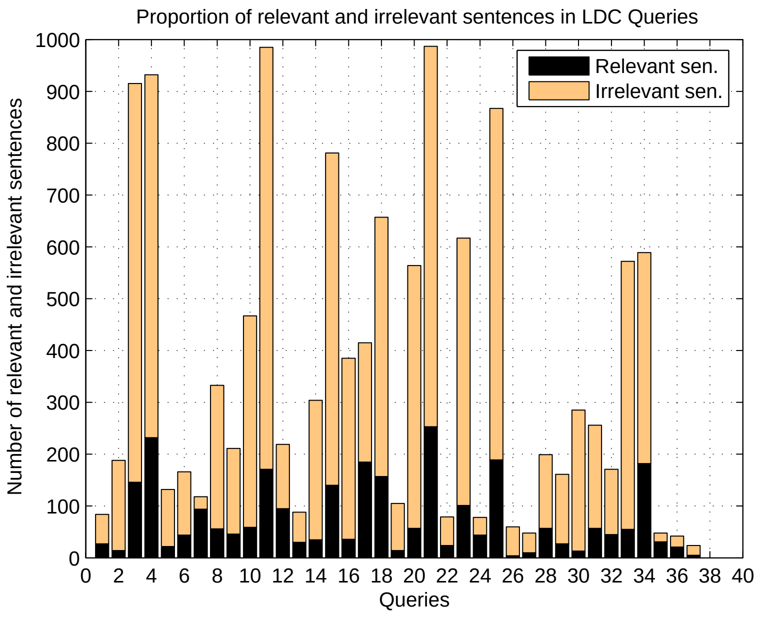


FIG. 5.1: The proportion of relevant sentences to irrelevant ones in 37 queries provided by LDC.

5.1.2 INDRI Documents

Another source of documents employed for distillation tasks is INDRI Information Retrieval System from UMass. INDRI employs an inference network approach, combining multiple evidences of relevance using statistical models. Therefore, the returned documents are associated with their own relevance scores. Similarly, the arguments in the GALE distillation queries are represented as nodes in the inference network and their appearances in the documents are scored with argument scores (Hakkani-Tür *et al.*, 2007).

The collection of documents provided by INDRI for a particular query of $template_i$, are not necessarily same as those provided by LDC, but may have some overlaps.

The proportion of relevant and irrelevant sentences per query are shown in 5.2.

5.2. Sentence Annotations

Usually the sentences inside the documents should be annotated manually. The annotation guidelines of GALE, give the necessary instructions to determine whether a sentence is relevant or not. The following text is extracted from "Annotation Guidelines for Distillation Training Data" prepared by ACE. It explains how template1 queries should be annotated.

"Relevant information focuses on the event and the persons, places, and activities directly associated with the event. Relevant information also includes subevents, causes, goals, and precursor or preparation events.

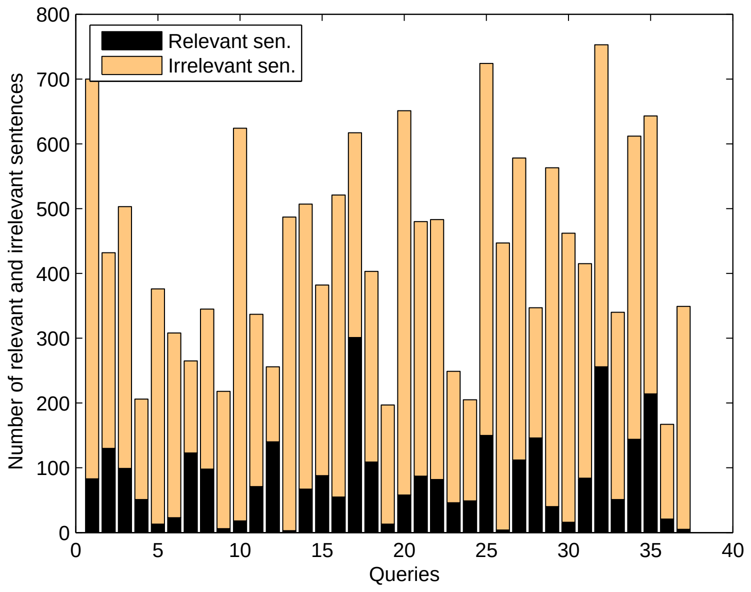


FIG. 5.2: The proportion of relevant sentences to irrelevant ones in 37 queries provided by INDRI.

For persons associated with the event: relevant information includes their role and the reason for their involvement in the event, as well as their involvement in promoting, funding, or planning for the event. Information about a person must be directly related to the event if it is to be relevant. Involvement in the event may be confirmed or suspected. Direct reactions, direct consequences, and the significance of the event are also relevant. For locations associated with events: only information that directly pertains to the even is relevant."

5.3. Features

When the input data to an algorithm is too large to be processed and it is suspected to contain much data, but not necessarily much information), then the input data should be transformed into a reduced representation set of features (also named features vector). Transforming the input data into the set of features is called features extraction. If the features extracted are carefully chosen it is expected that the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input (Gorgel & O.N.Ukan, 2007).

For example, the performance of the classifier can be improved by incorporating statistical information at the word sequence level such as selection of higher rank words. In the context of this work, rank is equal to frequency of occurrence of a word in text when all stop words have been removed. Various methods can be used to incorporate word history or modifying word neighborhood. Modification includes proposing new word candidates such as the stem of word or synonyms of words. By word stemming or synonyms, we can expand the coverage of each feature .

Throughout this work, we have used n-grams from word transcriptions as classification features. We have also included stemming and synonyms to cover more sentences as relevant ones.

5.4. Synonyms

Among 17 query templates defined in GALE framework, template1 queries can be considered as the least trainable queries, since they are open-domain queries and all the data is concentrated in slot. Therefore, to be able to find a correct answer to these types of queries, we need to collect as much data as possible related to query slot.

In our unsupervised approach, the decision of relevance to a particular query, depends either on the number of occurrences (frequency) of query components in documents or on the TD/IDF coefficient for that component. Talking about query components, we mean word botth transcriptions and their synonyms. Then for each word, we take a collection of synonyms (2 per word) and compute frequencies for the words in this new set. Depending on these frequencies, we keep some of the synonyms and throw out the rest. The final set of words, is a sorted list of high frequency query words and their synonyms. During sentence extraction, they will be used either as unigrams or as bigrams. Longer N-grams (more than 2) do not bring significant change to classification performance.

5.5. Parameters and Thresholds

To improve the classification quality, we need to define a set of thresholds and parameters in our algorithms. As it is usually done in classification approaches, several thresholds for the classifier are used, until the best performance on training set is achieved. This particular set of thresholds is applied to the classifier when it is running on test set.

In this work we have 5 types of thresholds:

- Mean Threshold: This parameter leaves a margin below mean base level of frequency, of word occurrences and lets choose words with frequencies bigger the margined mean. This value ranges from 0 to 0.4.
- TF-IDF Threshold: Same as Mean Threshold, just leaves a margin below TF-IDF base level. This threshold changes between 0 and 0.4.
- Number of iterations of classifier. The classification is done either at 100 or at 200 iterations.
- Number of iterations of algorithm. This value changes between 1 and 3.
- Probability threshold: The final output of the boosting algorithm are weights of the label l, attributed to a sample x. The weights f(x, l) need to be interpreted and mapped to a class. The conversion from the weights output by the Boosting to the attribution of a class is done as described below.
  Each weight is turned into probability $p(l|x)$ according to the following equation:

  $$p(l|x) = \frac{1}{1 + e^{-2f(x,l)T}} \tag{5.1}$$

  where T is the number of iterations used to train the classifier and $f(x,l)$ the final classifier output by the learning algorithm.

This probability, which is in fact the probability of correctness of sample x, forms our last threshold. For example by setting the probability threshold equal to 0.3, we consider samples with probability bigger than 0.7 as relevant and those with probability less than 0.3 as irrelevant. This variable changes from 0 through 0.4.

	Prior distribution of classes	Classifier estimation
True positive(TP)	Relevant	Relevant
True negative(TN)	Irrelevant	Irrelevant
False positive(FP)	Irrelevant	Relevant
False negative(FN)	Relevant	Irrelevant

FIG. 5.3: The four possible combinations between prior distribution of classes and classifier estimation.

5.6. Performance factors

To estimate the accuracy of a classification, we compare for each instance in the test set the class estimated by the classifier to the one assigned by humans, which we refer to as the correct one. The labeling done by humans is thus considered as the absolute truth when evaluating the hypotheses done by a classifier, even though there may be mistakes in them comparing to annotation guideline.

There exist four designations to compare the estimation of the system with the correct classification: the true positives/negatives and the false positives/negatives. The true positives and the true negatives represent the examples that the classifier guess for them is correct. More precisely, if a sentence is identified as relevant by the classifier and it is relevant in reality as well, it is considered as *true positive* and a sentence that the classifier identifies as irrelevant correctly, will be interpreted as *true negative*. As can be observed in 5.1 and 5.2, the irrelevant sentences are much more frequent in documents and that happens due to open-domain property of template1 queries. The two remaining classes are the ones containing the examples that were wrongly classified. The *false negatives* represent the irrelevant sentences identified by classifier, whereas it is relevant in reality. Finally, the *false positives* are the irrelevant sentences which are mal-estimated by classifier as relevant. The designations are summarized in 5.3.

Since the whole distillation task is now framed as a classification problem, to measure the performance of the system as a concrete number, we used primarily *classification error rate* and then *F-measure*.To do so, the classifier's output scores should be interpreted as binary values and turn into binary decisions. Depending of correctness or incorrectness of decisions, TP, TN, FP and FN values are determined. Then by using these 4 designations, we compute the standard recall and precision metrics to compute F-measure as a performance measure of our system. Once these values obtained, we perform macro-averaging over queries since the number of relevant snippets per query varies significantly. We have performed n-fold cross-validation where in each fold one query was used for testing, and the rest for training. The classifier output is then used to compute the error rate and F-measure values for various queries.

5.6.1 Classification Error Rate (CER)

Error rate or test error is defined as the number of errors done in the classification, divided by the total number of instances on which the classifier was tested:

$$\text{Classification Error Rate} = \frac{FN + FP}{FN + FP + TP + TN}.$$

5.6.2 F-measure

In order to understand what the F-measure is, one first has to understand two intermediate measures, the *recall* and the *precision*.

Recall

The recall is the ratio of relevant sentences that are retrieved, out of all true labeled sentences(Relevant and irrelevant). It can be expressed as:

$$\text{Recall} = \frac{TP}{TP + FN}. \tag{5.2}$$

Hence, if only recall was taken into account to measure the performance, the best solution would be to label every sentence as relevant, which would yield to a recall of 100%. Such a case is avoided by combining the recall measure with the precision measure.

Precision

Precision is the ratio of retrieved and pertinent(relevant) sentences to all relevant sentences.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5.3}$$

In the previous extreme case with a recall of 100%, the precision would thus be very low. On the contrary, if only non-relevant sentences are returned the precision is 100%, but the recall is 0%.

Recall and precision measure the extent to which the system produces all the appropriate output (recall) and only the appropriate output (precision).

The recall and precision measures taken separately are however not sufficient, as pointed out by the above-mentioned extreme cases. These two measures are complementary and should therefore be combined. This is achieved by the F-measure, which is nothing but the weighted harmonic mean of recall (R) and precision (P) measures:

$$\text{F-Measure} = \frac{P \cdot R(\alpha + 1)}{\alpha P + R}$$

where α is a coefficient usually set to 1.

5.7. Baseline

Theoretically, baseline consists of information gathered at the beginning of the study from which, variations found in the study are measured. Deciding which is an appropriate baseline is a difficult task, however in the case of classification, usually chance baseline is used. We then compute the measures on unseen data using the proportion of observed agreement with baseline (Fazly *et al.*, 2006). In a chance baseline, all the sentence are labeled randomly. In the framework of classification, random labeling has a standard definition. It is equivalent to labeling all the sentences as relevant or all as irrelevant.

To compare the error rate improvement, we use the baseline with all irrelevant sentences. It looks a reasonable choice, since in our corpus the majority of sentences are irrelevant to their query. Though to have least error for baseline, we need to label all the sentences as 0. This baseline is called chance0 baseline. More clearly, we find the performance measure in the case that every sentence are labeled with equal chance to a single class (here is 0).

This type of labeling of baseline will lead to have the recall equal to zero. Thus, the F-measure would be equal to zero for all the queries of baseline. In order to resolve this problem, in the case of F-measure comparison, we choose the other type of baseline. We label all the sentences of training set as 1. This baseline is called Chance1 baseline. Again we compute the F-measure of classifier over our test set, and compute the improvement to baseline results.

Table 5.1, shows a resume about two individual parts of our corpus corresponding to LDC Documents:

	Queries from GALE-Y1	**Queries from GALE-Y2**	**Total**
Documents	526	97	623
Sentences	10785	2347	13132

Table 5.1: Summary of LDC Corpus Statistics.

The same, table 5.2, shows same type of information about INDRI corpora:

	Queries from GALE-Y1	**Queries from GALE-Y2**	**Total**
Documents	535	200	735
Sentences	11501	4651	16152

Table 5.2: Summary of INDRI Corpus Statistics.

To explain the computation of baseline measures, consider a query with following prior information:

$$\begin{aligned} \text{\# of Rel. Sentences} &= 46 \\ \text{\# of Irrel. Sentences} &= 165 \end{aligned}$$

The error and F-measure can be easily computed as following:

$$\begin{aligned} \text{Chance0 Error} &= \frac{P}{P+N} = 0.2180 \\ \text{Chance1 Precision} &= \frac{P}{P+N} \\ \text{Chance1 Recall} &= 1 \\ \text{Chance1 F-measure} &= \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2 * P}{2 * P + N} = 0.3580 \end{aligned}$$

where P stands for relevant sentences and N stands for irrelevant ones.

5.8. Statistical Significance Test

Test of *statistical significance* addresses the question, how likely is a result, assuming the null hypotheses to be true (Shaver, 1993). A statistically significant difference, means there is statistical evidence that there is a difference; it does not mean the difference is necessarily large, important or significant in the usual sense of the word. The most common level, used to mean something is good enough to be believed, is 0.95. This means that the finding has a 95% chance of being true. For example, a value of "0.01" means that there is a 99% (1-.01=.99) chance of it being true.

The *significance level* is the probability that the null hypothesis will be rejected in error when it is true (a decision known as a Type I error, or "false positive") (Hand *et al.*, 2001).

In this work, we have used a script written by Jeff Bilmes at ICSI. Assuming we have two methods (Our Method and Baseline), each run on N independent trials (Number of sentences to be classified), and p1 and p2 are two processes to be compared in terms of classification error rate.

"We want to find out if p2 is significantly (in a statistical sense) better than p1 under N samples. We do this by assuming a null hypothesis, H0, and seeing how unlikely this null hypothesis is.

Since we want to test whether one process (p2) is better than another p1 (as apposed to testing whether one process is either better or worse than another), we use the two hypothesis H0 and H1 as illustrated below.

```
H0:  p1 == p2, i.e.  the scores are really identical
H1:  p1 > p2 , i.e.  p2 is indeed better than p1 at the current significance level.
```

We do this by assuming H0 and then disproving it at various significance levels."

CHAPTER
6

Experiments and Results

Experimental results are presented in this chapter. In unsupervised methods, classification is performed in two phases. In the first one, only LDC corpus is used, whereas in second part INDRI documents are tested to be classified. For each experiment, several result sets are presented and compared with the baseline results. At each case, relative improvement compared to baseline and the result of statistical significance test are computed.

6.1. Phase I: LDC Data

In all of our experiments, we split the data to form a training set, a test set and a development (or held-out) set, as it is always done in classification tasks. The training set contains approximately 80% of the labeled data whereas the remaining 20% is evenly distributed between the test set and the development set.

The classifier learning process is done using the training set, the held-out set is optionally used to optimize parameters and thresholds and the test set is used to evaluate the performance of the classifier. In our experiments, we tried to optimize the acceptance threshold over the posterior probabilities for two classes irrelevant (0) and relevant (1). We generated the posterior probabilities $p(0|example)$ and $p(1|example)$ and computed F-measure and error rate by using different thresholds until we obtained the best performance over the development set. Then we applied that threshold on the test set and measured the classifier performance. In fact, $p(0|example)$ and $p(1|example)$ show the probability of correctness of distinguished sentence class, by the classifier. The posterior probabilities are calculated as shown in the previous chapter.

Test set should always be unseen, meaning that no parameters can be modified according to it. Also, it is expected that each classifier which is built on a specific corpus, performs the best when evaluated exactly on the very same corpus. Since both the training and test data share the same settings (query templates, annotation style). This is because of the iid (identical and independent distribution of examples from the same source) assumption of classifiers.

Cross-validation is a standard practice of partitioning a sample of data into subsets such that the analysis is initially performed on a single subset while the other subsets are retained for subsequent use in validating the initial analysis. The initial subset of data is the so called training set; the other subsets are called validation or testing sets. If the data size is not big enough, *n-fold cross validation* should be applied. This method helps to have a larger corpus for training data, and consequently more precise results on the test set will be produced. In n-fold cross-validation, the original corpus which consists of training and test data, is partitioned into n subsets. Among these n subsets, a single one is retained as the validation data for testing the model, and the remaining $(n-1)$ sets are used as training data. The cross-validation process is then repeated n times (the folds), with each of the n subsets used exactly once as the validation data. The n results from the folds then can be averaged to produce a single estimation.

To compare the performance of the proposed methods, we performed n-fold cross validation and computed the mean of the desired metric (classification error or F-measure) to compare to the baseline performance. In our experiments, we have 27 template 1 queries as described in the previous chapter.

By applying the proposed methods, we obtain two important sets of results. One set corresponds to the performance of classifier for test queries in terms of classification error rate and the other one corresponds to its performance in terms of F-measure.

As mentioned in previous chapter, we use a combination of parameters (thresholds) for each run of classification. Once we train the classifier, it is reasonable to find the set of parameters that lead to the best results on training set, and apply the same set of thresholds on test set and measure the performance.

In the case of n-fold cross validation, at each fold, for each parameter set, we sum up the desired metric (classification error rate or F-measure) over all the training data and take an average. Then the parameter set that leads to best average result (minimum error rate or maximum F-measure) on the held-out parameter optimization set is applied on test set.

6.1.1 Supervised Classification

As the first step, we started with the experiments using the supervised learning method. This was done in order to get the performance of the existing IXIR system for the template 1 queries.

The results of this experiment are shown in figure 6.1. The plots show lots of fluctuations for F-measure and error for different test queries, which is actually a good sign for the necessity of a fair evaluation. In some cases we observe an F-measure equal to zero. It comes from the fact that classifier has recognized no relevant examples for that query. In other words, it was unable to learn correct words on the existing training set. It happens due to the generality of template 1 queries.

The ultimate goal of supervised classification baseline is having an average of the measured metric for all test queries. In this experiment, the average for F-measure is around 12.52 and the average for the error rate is 24.38. It is obvious that the very low F-measure comes from the effect of zero value F-measures for some test queries. Note that, when we use a trivial classifier which selects all sentences as relevant, the F-measure is much higher, proving that the existing data-driven IXIR approach is unsuitable for this template. It is also intuitive since it is not clear what kind of features the classifier would learn. This happens since the query slots can be anything in this template and there is no pattern.

Figure 6.2, shows the diagrams for Precision and Recall at different runs. In some of the cases both precision and recall are equal to zero. These are the same points that number of relevant sentences recognized by classifier are equal to zero. Again it proves that the raw method of classification and query answering in template 1, needs to be enhanced. In the areas that they are not zero, we observe a trade-off between precision and recall. When we get a very good precision, the recall drops or vice versa.

Table 6.1 shows a comparison between the measured metrics of n-fold with those of baseline when classification is done in a supervised fashion using only LDC data.

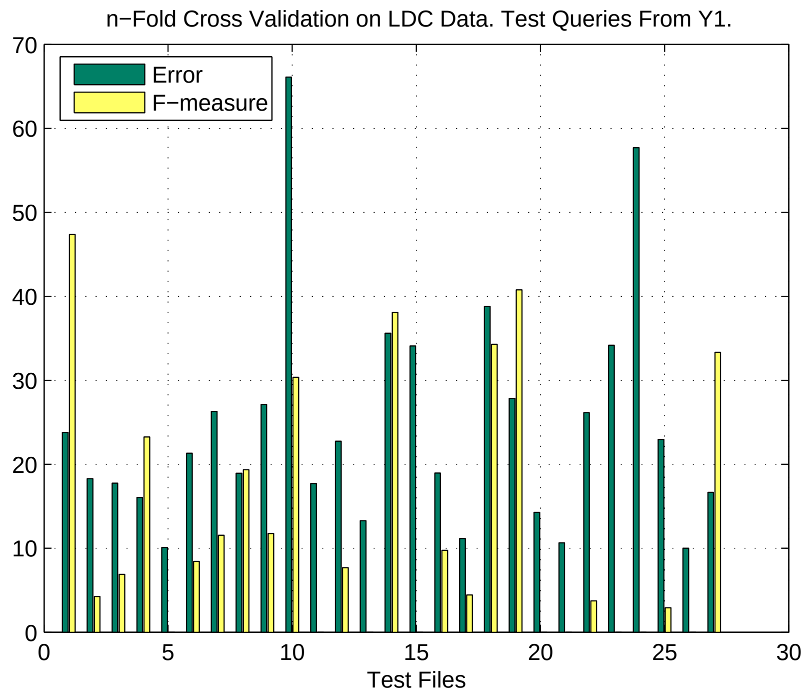


FIG. 6.1: F-measure and error rate of 27 queries of LDC data in n-fold cross validation experiment for **supervised** classification.

	Classification by Words	**Baseline**	**Relative Improvement %**
Error rate	24.38	25.60	4.7
F-measure	12.52	38.34	-67.34

Table 6.1: Performance of the **supervised** classification versus baseline using n-fold cross validation. The training corpus comes from **LDC** data.

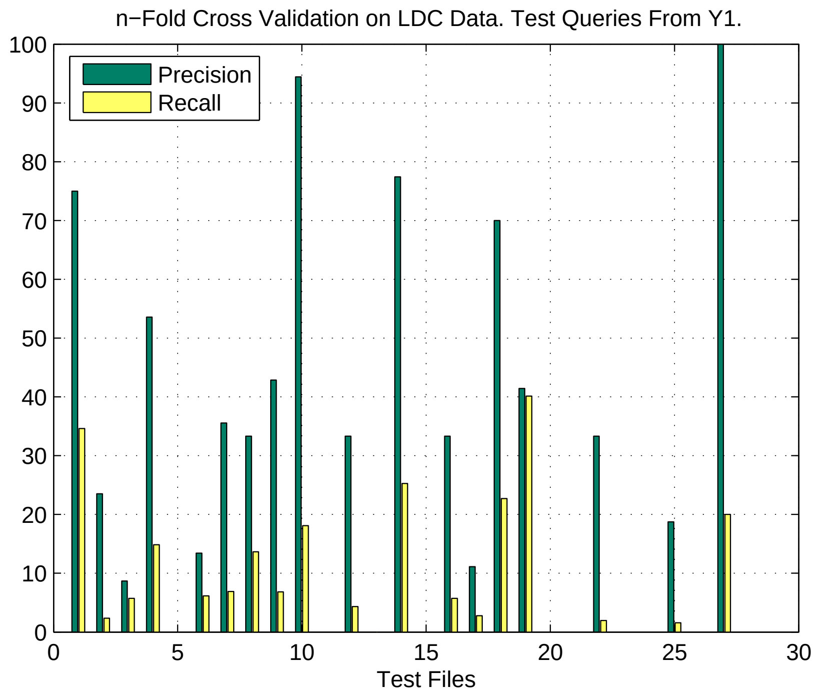


FIG. 6.2: Precision and Recall of 27 test queries in n-fold cross validation experiment for LDC data for **supervised** classification.

6.1.2 Unsupervised Classification

In this section, we present our results using the proposed unsupervised learning methods for information distillation. For each method, we provide query by query error rate and F-measure figures and the averages. The number of iterations is optimized using the held-out set. The maximum of this value is set to 3 iterations.

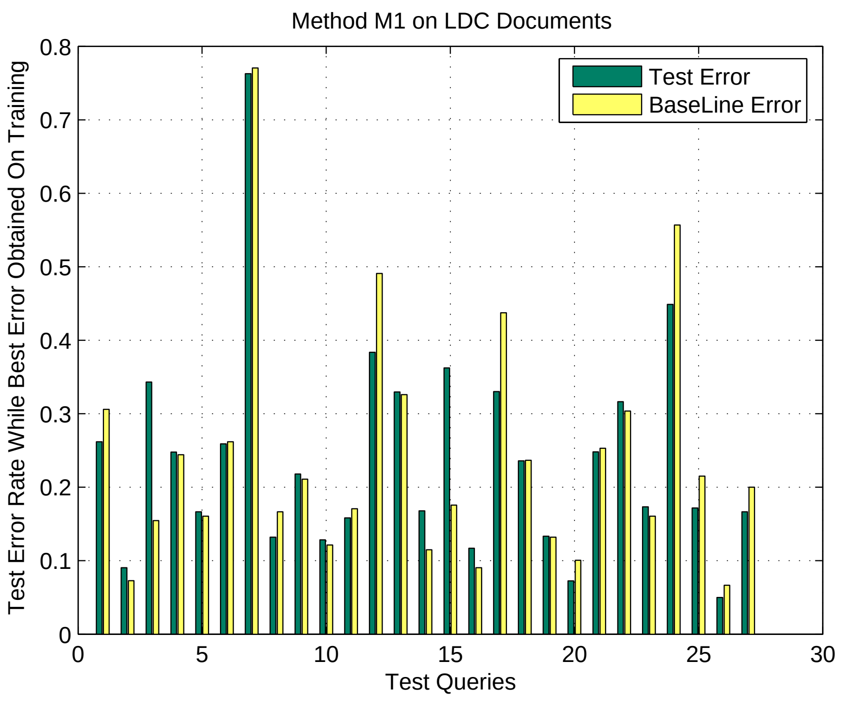
M1: Strict Initial Classification For Relevant Snippets

The result sets can be observed in figure 6.3. Table 6.2 shows a numeric comparison of classification error rate or F-measure changes, between classification results and the baseline. When we measure the test error of each query by this method, in 18 cases we obtain better results than baseline whereas in 9 case the classifier acts weaker than baseline.

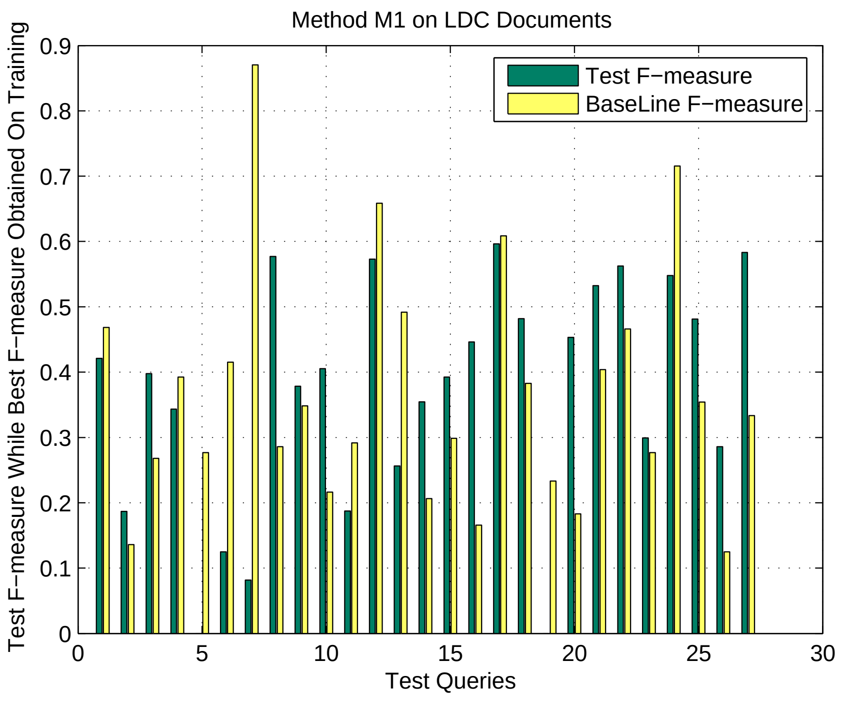
	Improvement	**No Changes**	**Degradation**
Error	14	0	13
F-measure	16	0	11

Table 6.2: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M1** on **LDC** documents corresponding to GALE-Y1 queries.

In average, as can be observed in table 6.3, Method M1 decreases the classification error rate by 0.38% and increases the F-measure by 0.77%.



(a) Test error rate of Method M1 Under Best Results For Training Set



(b) Test F-measure of Method M1 Under Best Results For Training Set

FIG. 6.3: **Test error rate** and **F-measure** while having best training classification error when method **M1** is executed on **LDC** documents corresponding to GALE-Y1 queries.

	Method M1	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2399	0.2408	0.3881	–
F-measure	0.3685	0.3657	0.7775	–

Table 6.3: Average classifier performance versus average baseline performance, while having the best results for training set in method **M1** on **LDC** documents corresponding to GALE-Y1 queries.

M2: Strict Initial Classification For Relevant and Irrelevant Snippets

The result sets can be observed in figure 6.4. As before, table 6.4 shows a numeric comparison of error rate or F-measure changes, between test results and baseline while having best training results.

	Improvement	**No Changes**	**Degradation**
Error rate	14	0	13
F-measure	17	0	10

Table 6.4: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M2** on **LDC** documents corresponding to GALE-Y1 queries.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2424	0.2408	-0.6587	N/A
F-measure	0.3797	0.3657	**3.8313**	**0.05**

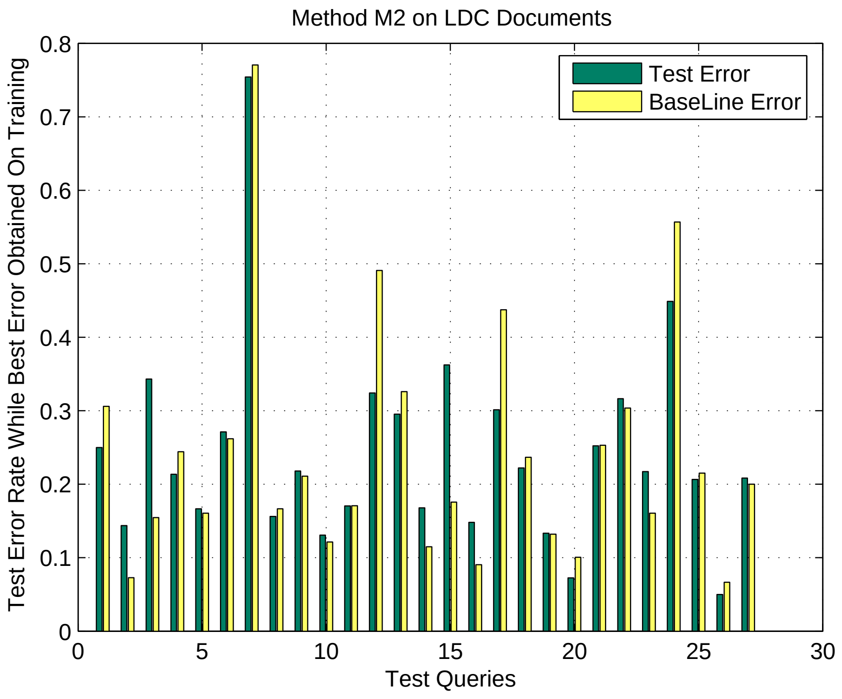
Table 6.5: Average classifier performance versus average baseline performance, while having the best results for training set in method **M2** on **LDC** documents corresponding to GALE-Y1 queries.

On average, as can be observed in table 6.5, Method M2 doesn't perform any improvements on classification error rate, but increases the F-measure by 3.83%. This improvement is actually better than the one obtained with the M1 method. The reason might be applying more strict criteria to find the irrelevant classes, comparing to M1. In this method, the relevancy was defined as presence of *all words* and irrelevancy by absence of *all words* passed through the mean threshold. Thus, there is probability to lose some of sentences which don't fall in either category and never be learned by classifier. However, this may result in less noisy training data. It means that in general, the accuracy of classified sentences as relevant, is quite high. The significance test shows that with a probability of 95% the difference obtained between F-measure of our study and baseline is not due to chance.

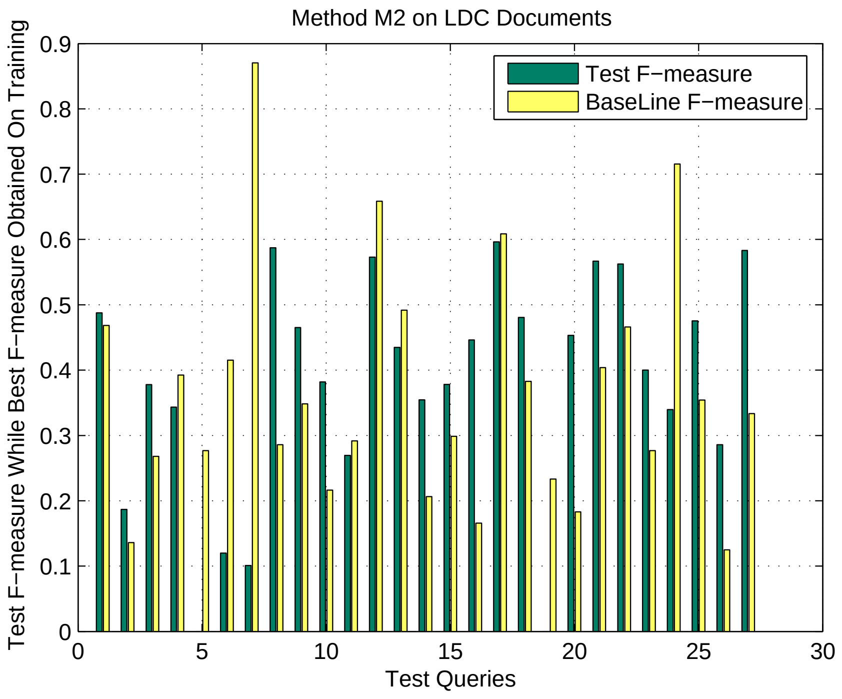
M3: Relaxation of Initial Classification For Relevant and Irrelevant Snippets

The result sets can be observed in figure 6.5. As before, table 6.6 shows a numeric comparison of test error or F-measure changes, between test results and baseline while having best training results.

In Method M3, the classification error rate is not reduced, but the F-measure is increased by 6.55%. The reason of degradation in CER might happen because of providing too much relaxation in this method. During different iterations, this method does not rely on the previous results of the classifier. That is, each run is completely independent from the previous one and doesn't get any feedback from previous iteration. For instance, we may look for 4 words in first run, then

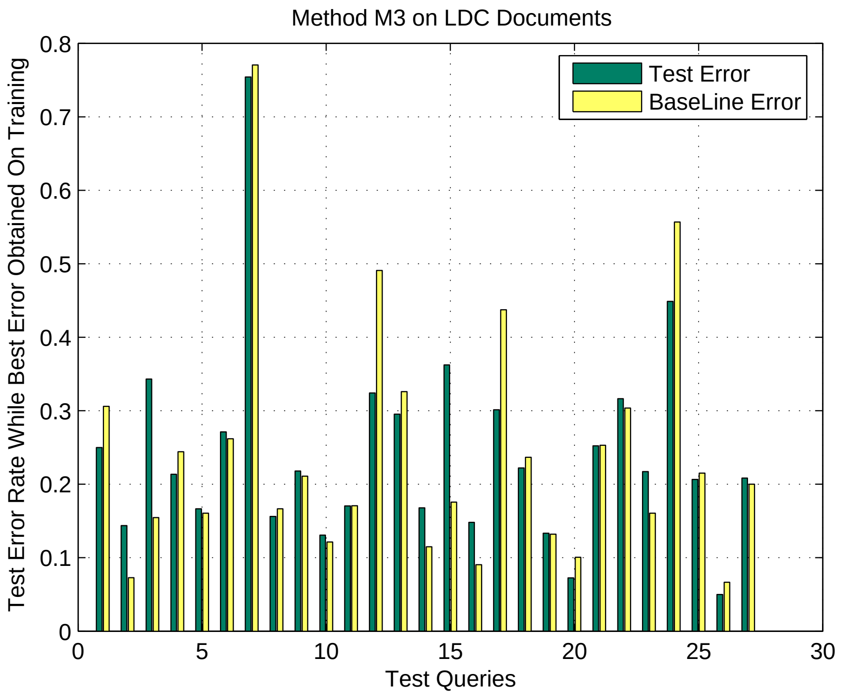


(a) Test error rate of Method M2 Under Best Results For Training Set

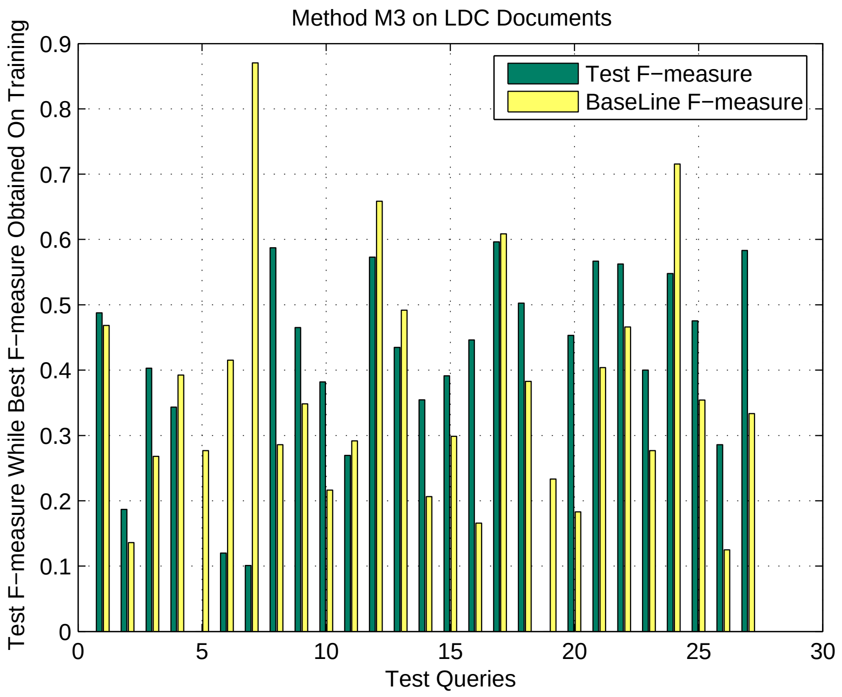


(b) Test F-measure of Method M2 Under Best Results For Training Set

FIG. 6.4: **Test error rate** and **F-measure** while having best training classification error when method **M2** is executed on **LDC** documents corresponding to GALE-Y1 queries.



(a) Test error rate of Method M2 Under Best Results For Training Set



(b) Test F-measure of Method M3 Under Best Results For Training Set

FIG. 6.5: **Test error rate** and **F-measure** while having best training classification error when method **M3** is executed on **LDC** documents corresponding to GALE-Y1 queries.

	Improvement	No Changes	Degradation
Error rate	14	0	13
F-measure	17	0	10

Table 6.6: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M3** on **LDC** documents corresponding to GALE-Y1 queries.

	Method M2	Baseline	Relative Improvement %	Stat. Sig. Test Level
Error rate	0.2424	0.2408	-0.6587	N/A
F-measure	0.3896	0.3657	6.5525	**0.001**

Table 6.7: Average classifier performance versus average baseline performance, while having the best results for training set in method **M3** on **LDC** documents corresponding to GALE-Y1 queries.

we look for 3 words in next run and so on. Since the search is reset at each time, it is possible that the number of errors increase significantly in iterative runs and that can be the reason for degradation of classification error rate. However, the F-measure has improved to an acceptable level. It means that in general, the accuracy of the classified sentences as relevant is quite high. The significance test shows that with a probability of 99.9% the difference obtained between F-measure of our study and baseline is not due to chance.

M4 Word Selection By TF-IDF At Document Level

The result sets can be observed in Figure 6.6. Table 6.8 shows a numeric comparison of error or F-measure changes, between test classification results and baseline while having best training results.

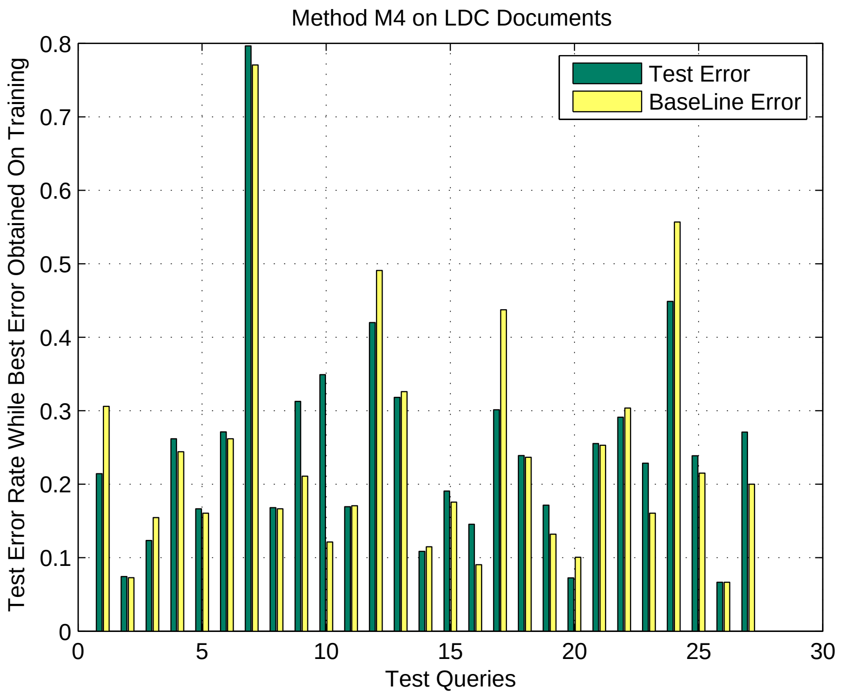
	Improvement	No Changes	Degradation
Error rate	10	0	17
F-measure	11	0	16

Table 6.8: This table shows number of cases that improvement or no change or degradation of classification error rate or F-measure has happened comparing to baseline, in method **M4** on **LDC** documents corresponding to GALE-Y1 queries.

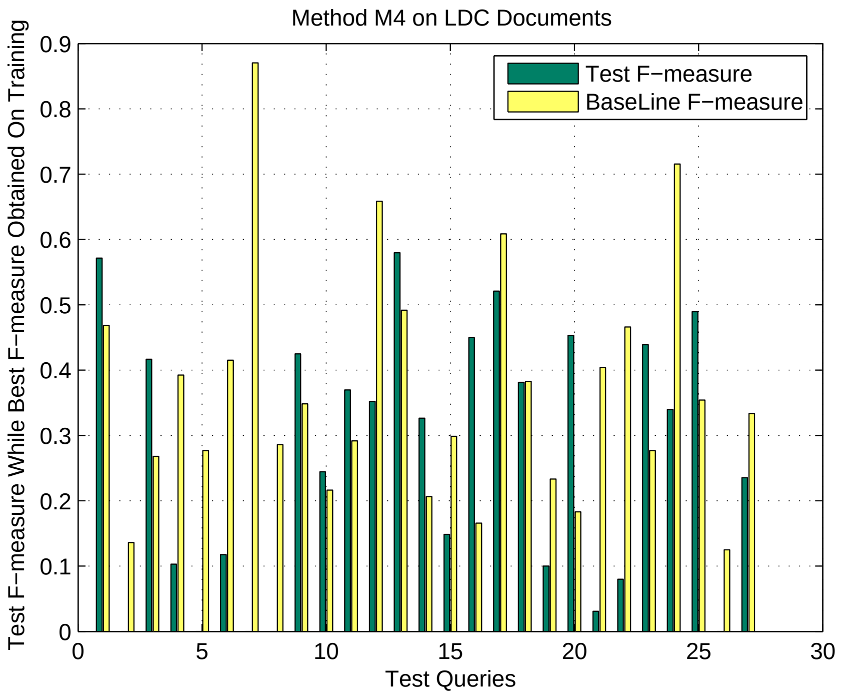
In Method M4, the classification error rate is not reduced. It means that extracting words with high TF-IDF and using them either as unigram or as bigram, does not help in terms of error rate. Also, we observe a degradation of F-measure which means that the choice of our features, is not good for precision and recall. Note that this is the F-measure of the positive class, as usually done in the information retrieval literature. Given that it is known that these documents have at least one relevant sentence, choosing all the sentences in the baseline guarantees 100% recall. Actually this is a tougher baseline to beat than another baseline, which would randomly assign x% of the sentences as relevant, where x% is the prior probability of being relevant.

M5 Word Selection By TF-IDF At Sentence Level

The result sets can be observed in figure 6.7. As before, table 6.10 shows a numeric comparison of error or F-measure changes, between test classification result and baseline while having best training results.

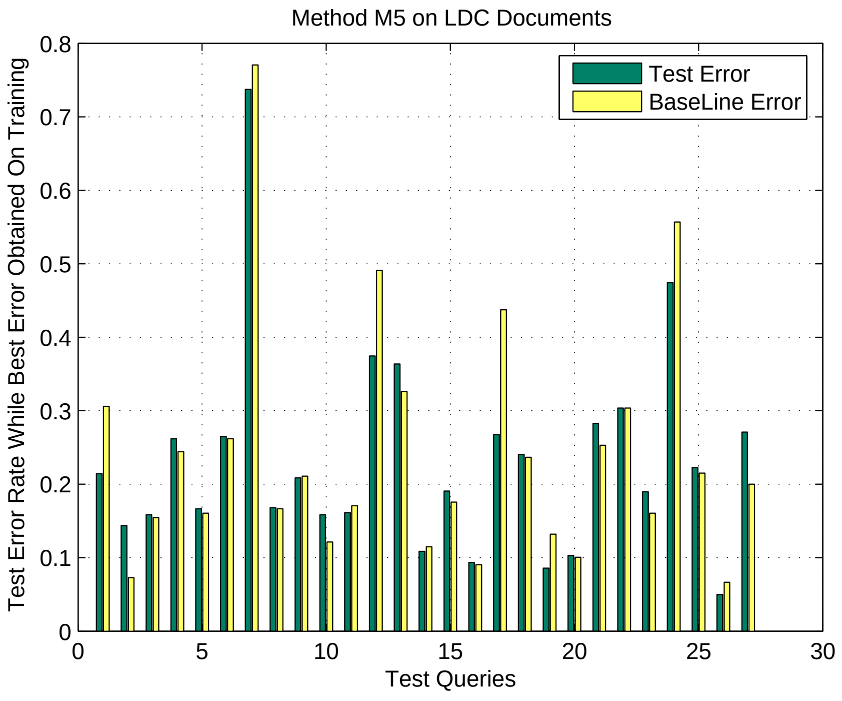


(a) Test error rate of Method M4 Under Best Results For Training Set

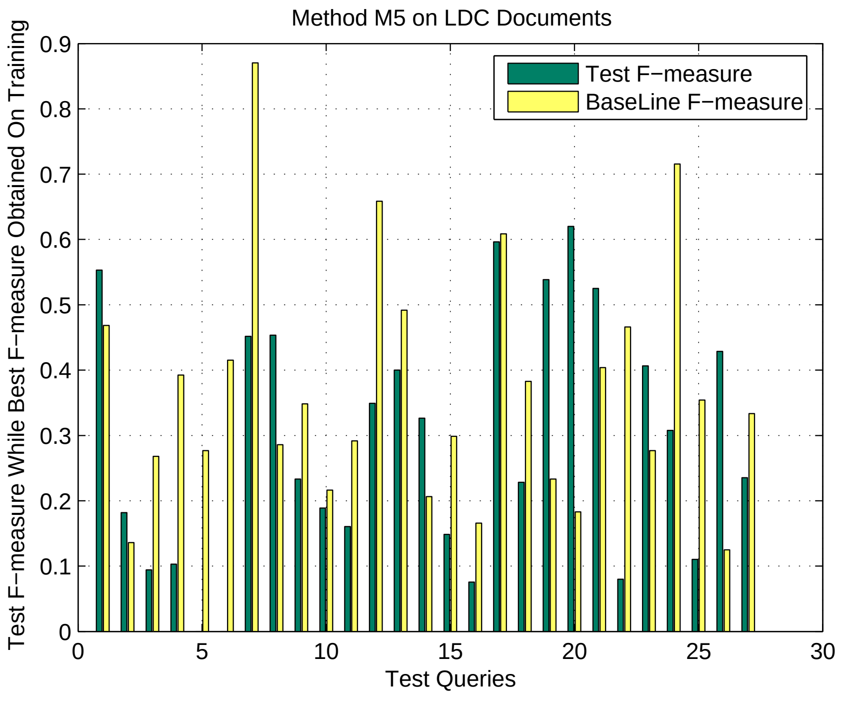


(b) Test F-measure of Method M4 Under Best Results For Training Set

FIG. 6.6: **Test error rate** and **F-measure** while having best training classification error when method **M4** is executed on **LDC** documents corresponding to GALE-Y1 queries.



(a) Test error rate of Method M5 Under Best Results For Training Set



(b) Test F-measure of Method M5 Under Best Results For Training Set

FIG. 6.7: **Test error rate** and **F-measure** while having best training classification error when method **M5** is executed on **LDC** documents corresponding to GALE-Y1 queries.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2472	0.2408	-2.6724	N/A
F-measure	0.3105	0.3695	-27.3366	N/A

Table 6.9: Average classifier performance versus average baseline performance, while having the best results for training set in method **M4** on **LDC** documents corresponding to GALE-Y1 queries.

	Improvement	**No Changes**	**Degradation**
Error rate	11	0	16
F-measure	9	0	18

Table 6.10: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M5** on **LDC** documents corresponding to GALE-Y1 queries.

In Method M5, the only difference is that TF-IDF is found at the sentence level. It means that each sentence is considered as a document to increase the number of documents of our corpus. An improvement on error rate can be observed. It means TF-IDF extracts more relevant words, when the number of documents are higher. This result is correct with probability of 90%. But, the same problem still exists with F-measure, and since the structure of this method is similar to previous one, the degradation of F-measure must come due to the same problem.

M6: All-Words Selection

The result sets can be observed in Figure 6.8. Also, table 6.12 shows a numeric comparison of error or F-measure changes, between test classification result and baseline while having best training results.

When we select test queries from Y1, a relative improvement of 5.30% is achieved comparing to baseline at it is significant at level 95%. This method actually is much similar to the method M2. We used bigrams and synonyms in addition to M2 to decrease its error rate, and it was successful. However these changes do not have a positive effect on F-measure. Similar to the previous methods, this method fails to find relevant sentences although it is good at finding irrelevant ones.

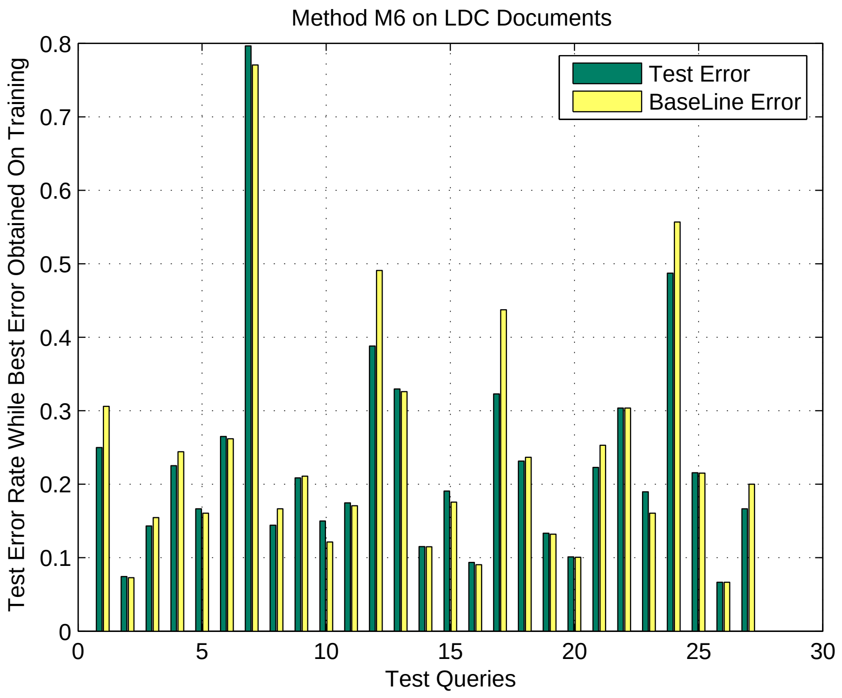
M7: Loose Word Selection

The result sets can be observed in Figure 6.9. Table 6.14 shows a numeric comparison of error or F-measure changes, between test classification results and baseline while having best training results.

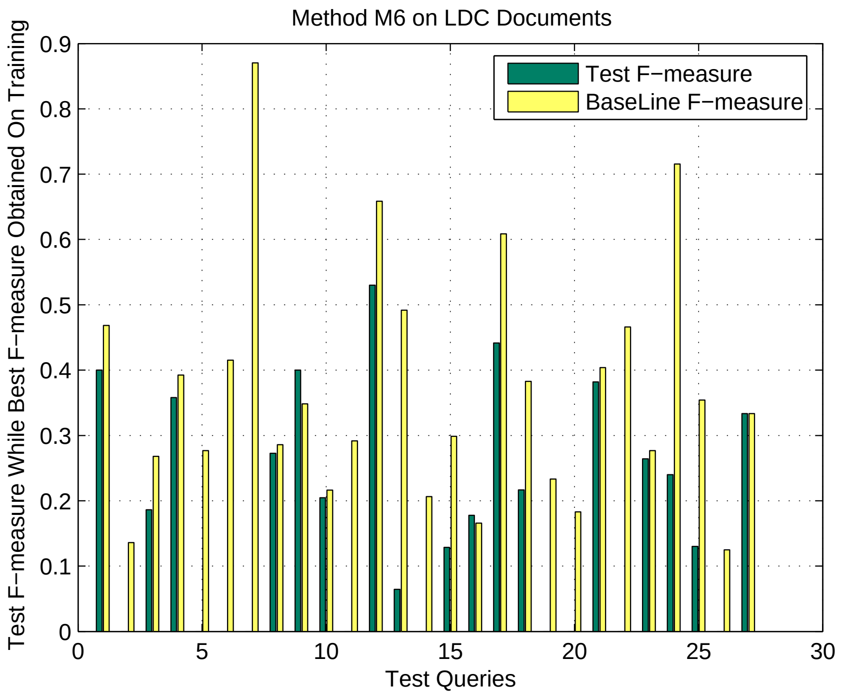
On GALE-Y1 data, comparing to other methods, this method has had the best performance in terms of CER and F-measure. The improvements on CER and F-measure are correct with prob-

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2320	0.2408	3.6367	0.1
F-measure	0.2887	0.3657	-21.0334	N/A

Table 6.11: Average classifier performance versus average baseline performance, while having the best results for training set in method **M5** on **LDC** documents corresponding to GALE-Y1 queries.



(a) Test error rate of Method M6 Under Best Results For Training Set



(b) Test F-measure of Method M6 Under Best Results For Training Set

FIG. 6.8: **Test error rate** and **F-measure** while having best training classification error when method **M6** is executed on **LDC** documents corresponding to **GALE-Y1** queries.

	Improvement	**No Changes**	**Degradation**
Error rate	12	0	15
F-measure	2	0	25

Table 6.12: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M6** on **LDC** documents. Test and training queries come from **GALE-Y1** source.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2280	0.2408	5.3031	**0.05**
F-measure	0.1752	0.3657	-52.0876	N/A

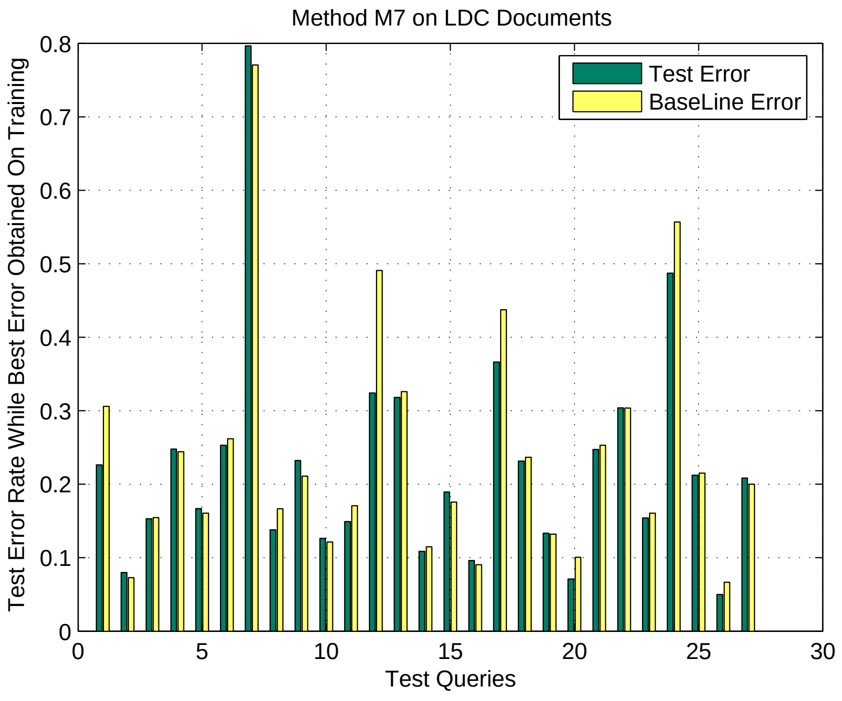
Table 6.13: Average classifier performance versus average baseline performance, while having the best results for training set in method **M6** on **LDC** documents. Test and training queries come from **GALE-Y1** source.

	Improvement	**No Changes**	**Degradation**
Error rate	16	0	11
F-measure	19	0	8

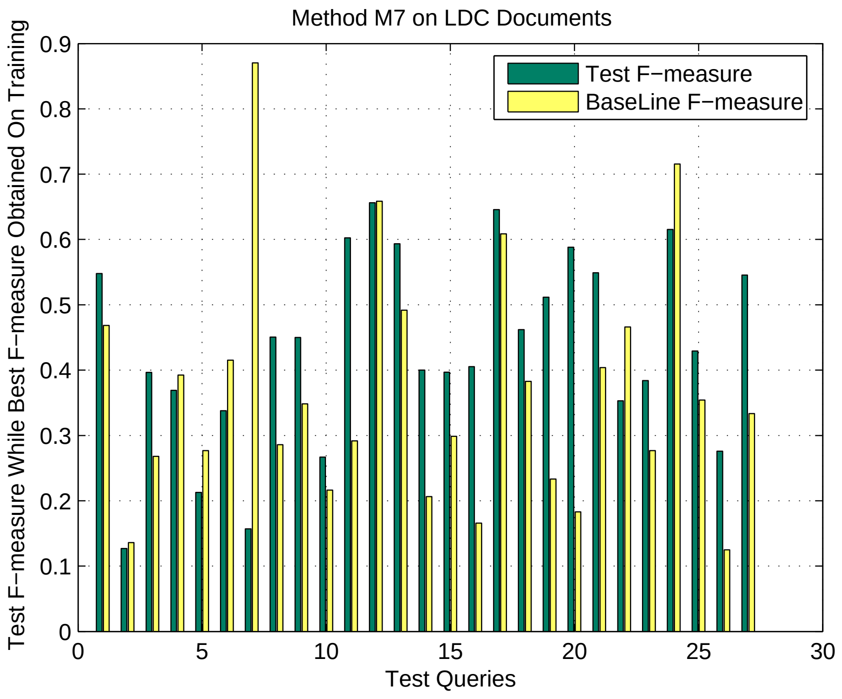
Table 6.14: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M7** on **LDC** documents. Test and training queries come from **GALE-Y1** source.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2248	0.2408	6.63	**0.005**
F-measure	0.4344	0.3657	**18.79**	**0.001**

Table 6.15: Average classifier performance versus average baseline performance, while having the best results for training set in method **M7** on **LDC** documents. Test and training queries come from **GALE-Y1** source.



(a) Test error rate of Method M7 Under Best Results For Training Set



(b) Test F-measure of Method M7 Under Best Results For Training Set

FIG. 6.9: **Test error rate** and **F-measure** while having best training classification error when method **M7** is executed on **LDC** documents corresponding to **GALE-Y1** queries.

ability of 99.5% and 99.9%, respectively. The strength of this method is that, it obtains at least one relevant sentence for each query. This way, it helps a considerable increase of F-measure. At the same time, the results show that this method detects irrelevant sentences at the correct extent which helps a reduction on error rate.

6.2. Phase II: INDRI Data

6.2.1 Unsupervised Classification

In this set of experiments, the same methods are run on the collection of documents returned by IR Engine of INDRI. The charts and set of information corresponding to the proposed 7 methods are presented below.

M1: Strict Initial Classification For Relevant Snippets

	Improvement	No Changes	Degradation
Error rate	11	0	16
F-measure	12	0	15

Table 6.16: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M1** on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M1	Baseline	Relative Improvement %	Stat. Sig. Test Level
Error rate	0.2018	0.1814	-11.2508	N/A
F-measure	0.2769	0.2867	-3.4466	N/A

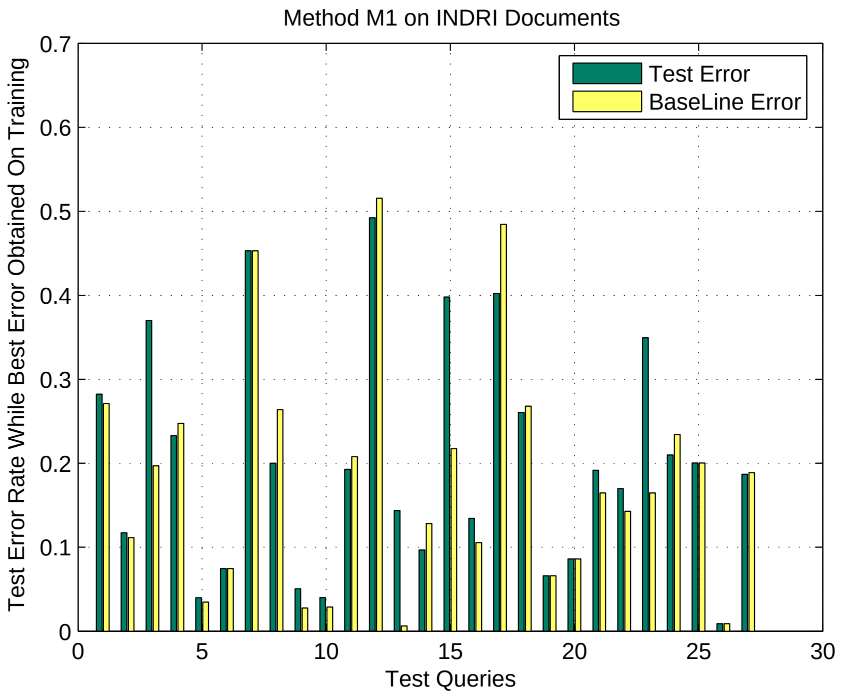
Table 6.17: Average classifier performance versus average baseline performance, while having the best results for training set in method **M1** on **INDRI** documents corresponding to GALE-Y1 queries.

This method performs worse in term of classification error rate and F-measure when compared to the results presented in the previous section using only the LDC data. The reason is that by using the LDC data, missing the small number of relevant sentences costs dearly for these parameters, however the setting in this experiment is closer to a real world scenario where some documents may have no relevant sentence.

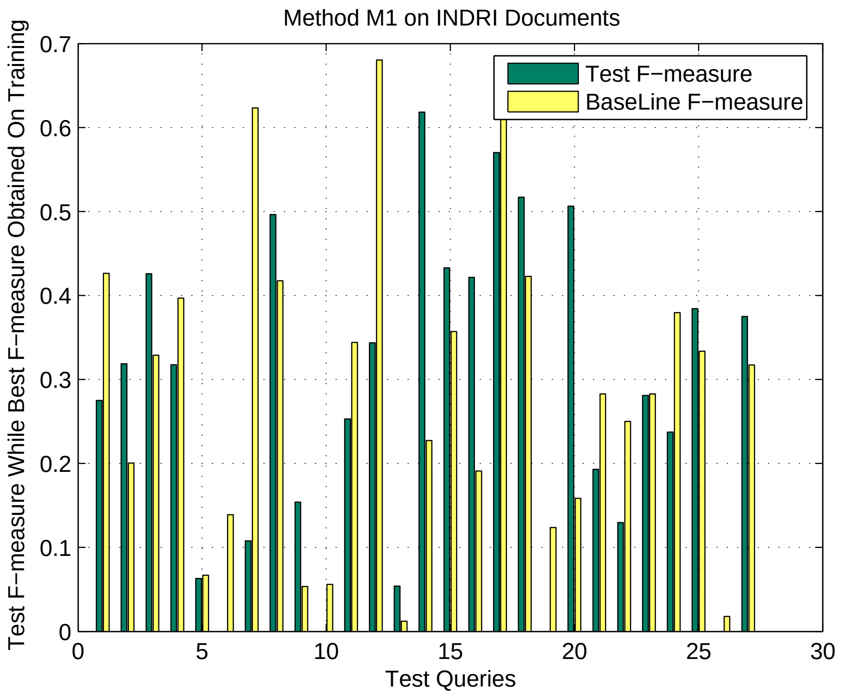
M2: Strict Initial Classification For Relevant and Irrelevant Snippets

The results of this experiment are reflected in figure 6.11 and tables 6.18 and 6.19.

Comparing to M1, this method performed much better in terms of F-measure improvements. The performance improves by more than 11% relative, showing the importance of having less noisy training data as in the previous section. The error rate did not decrease due to similar reasons discussed in the previous method.

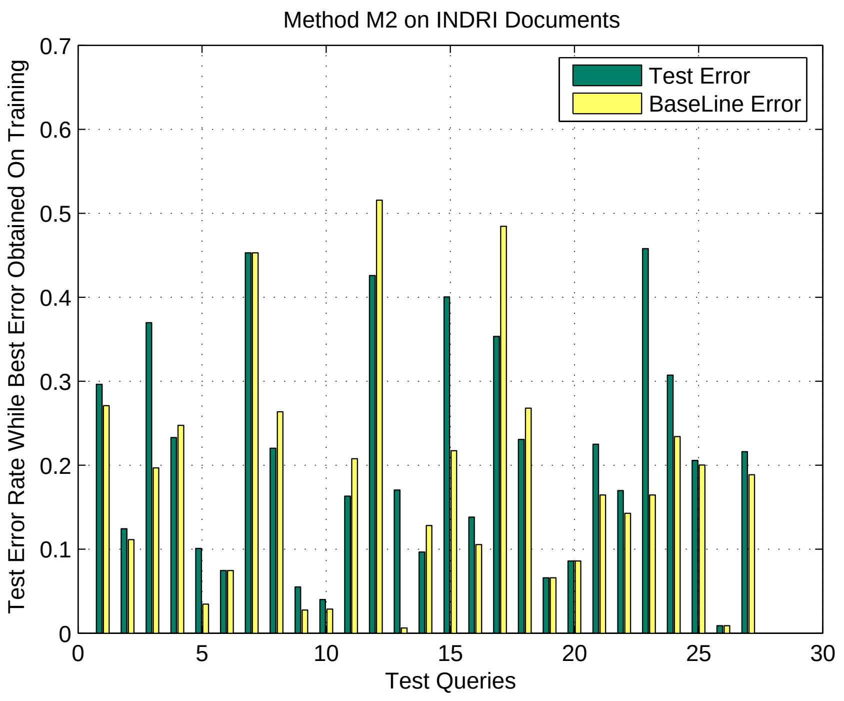


(a) Test error rate of Method M1 Under Best Results For Training Set

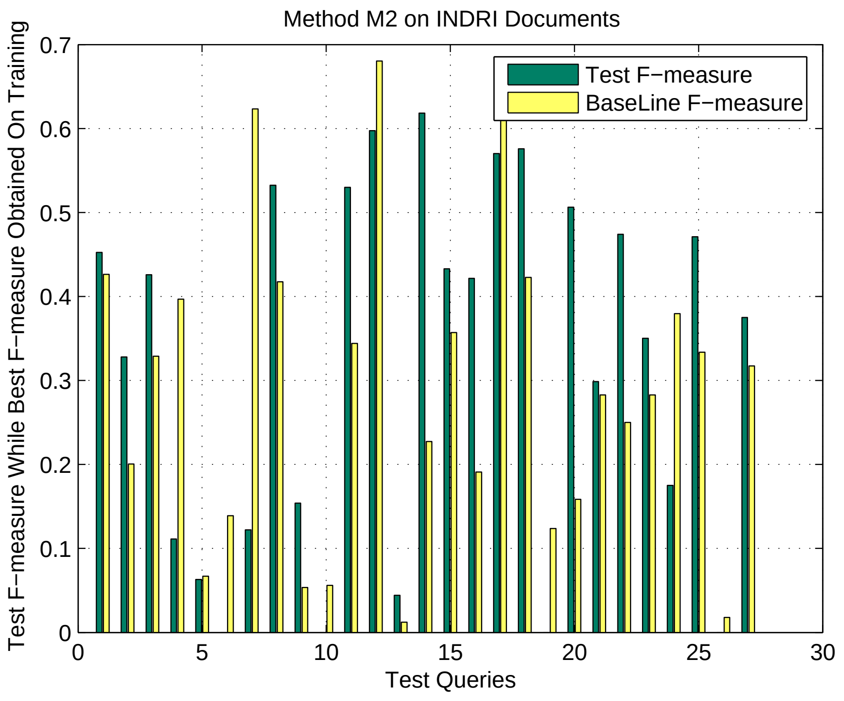


(b) Test F-measure of Method M1 Under Best Results For Training Set

FIG. 6.10: **Test error rate** and **F-measure** while having best training classification error when method **M1** is executed on **INDRI** documents corresponding to GALE-Y1 queries.



(a) Test error rate of Method M2 Under Best Results For Training Set



(b) Test F-measure of Method M2 Under Best Results For Training Set

FIG. 6.11: **Test error rate** and **F-measure** while having best training classification error when method **M2** is executed on **INDRI** documents corresponding to GALE-Y1 queries.

	Improvement	No Changes	Degradation
Error rate	9	0	18
F-measure	17	0	10

Table 6.18: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M2** on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M2	Baseline	Relative Improvement %	Stat. Sig. Test Level
Error rate	0.2107	0.1814	-16.1510	N/A
F-measure	0.3196	0.2867	11.4659	**0.001**

Table 6.19: Average classifier performance versus average baseline performance, while having the best results for training set in method **M2** on **INDRI** documents corresponding to GALE-Y1 queries.

M3: Relaxation of Initial Classification For Relevant and Irrelevant Snippets

Figure 6.12 shows the comparison of classification error rate and F-measure between our results and baseline. Tables 6.20 and 6.21 shows numerical results of comparison, as well.

	Improvement	No Changes	Degradation
Error rate	9	0	18
F-measure	16	0	11

Table 6.20: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M3** on **INDRI** documents corresponding to GALE-Y1 queries.

This method performed similarly to the previous one in general, even obtaining less improvement in F-measure, which shows it's capable of detecting relevant sentences correctly. However the error rate is increased. This shows the poor performance of it in detecting irrelevant sentences, hence providing higher false alarms.

M4: Word Selection By TF-IDF At Document Level

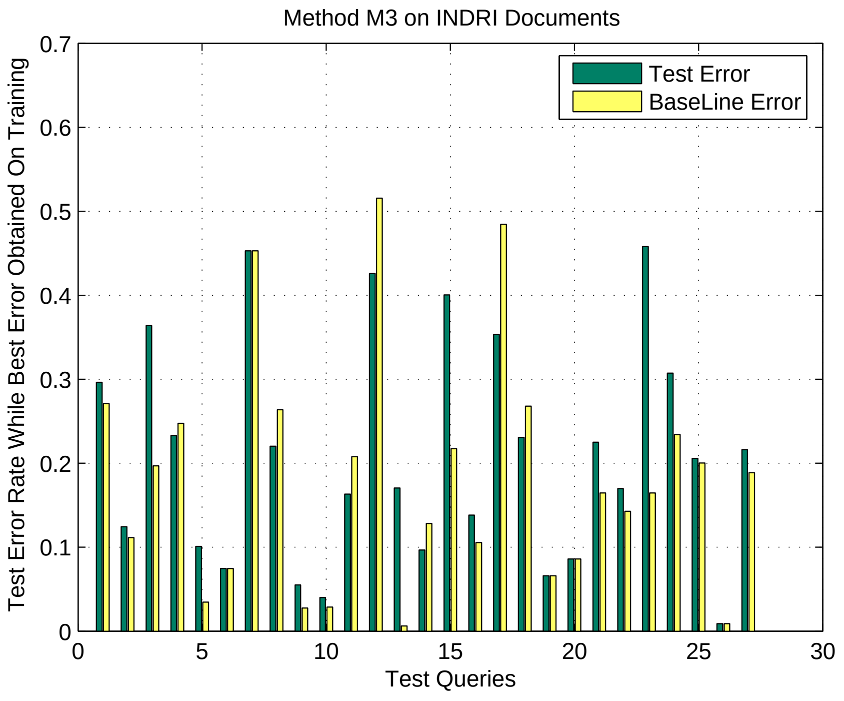
The results related to this method are shown at figure6.13 and tables 6.22 and 6.23.

Comparing to previous methods in the same category, this method cases a degradation in both CER and F-measure which shows TF-IDF coefficient is not necessarily a good criteria for word extraction.

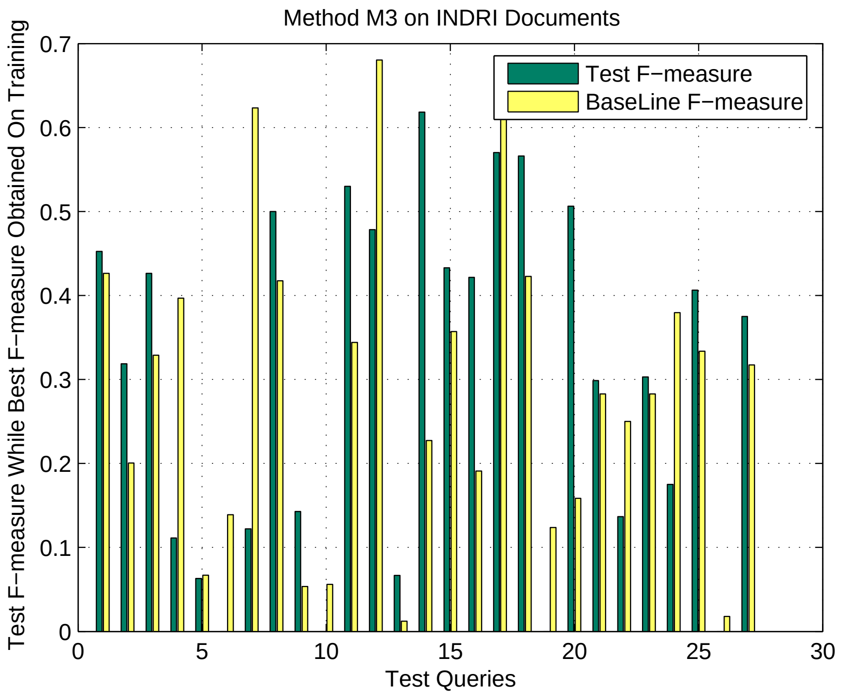
M5: Word Selection By TF-IDF At Sentence Level

The result sets can be observed in figure 6.14. As before, table 6.24 shows a numeric comparison of error or F-measure changes, between test classification result and baseline while having best training results.

Again, the same performance can be observed for a TF-IDF based method. By this observation, we believe that TF-IDF is not always a good criteria for word selection in real-world documents containing mostly irrelevant sentences.



(a) Test error rate of Method M3 Under Best Results For Training Set



(b) Test F-measure of Method M3 Under Best Results For Training Set

FIG. 6.12: **Test error rate** and **F-measure** while having best training classification error when method **M3** is executed on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2105	0.1814	-16.0292	N/A
F-measure	0.2971	0.2867	3.6145	**0.05**

Table 6.21: Average classifier performance versus average baseline performance, while having the best results for training set in method **M3** on **INDRI** documents corresponding to GALE-Y1 queries.

	Improvement	**No Changes**	**Degradation**
Error rate	11	0	16
F-measure	9	0	18

Table 6.22: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M4** on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.2030	0.1814	-11.9109	N/A
F-measure	0.2466	0.2867	-14.0078	N/A

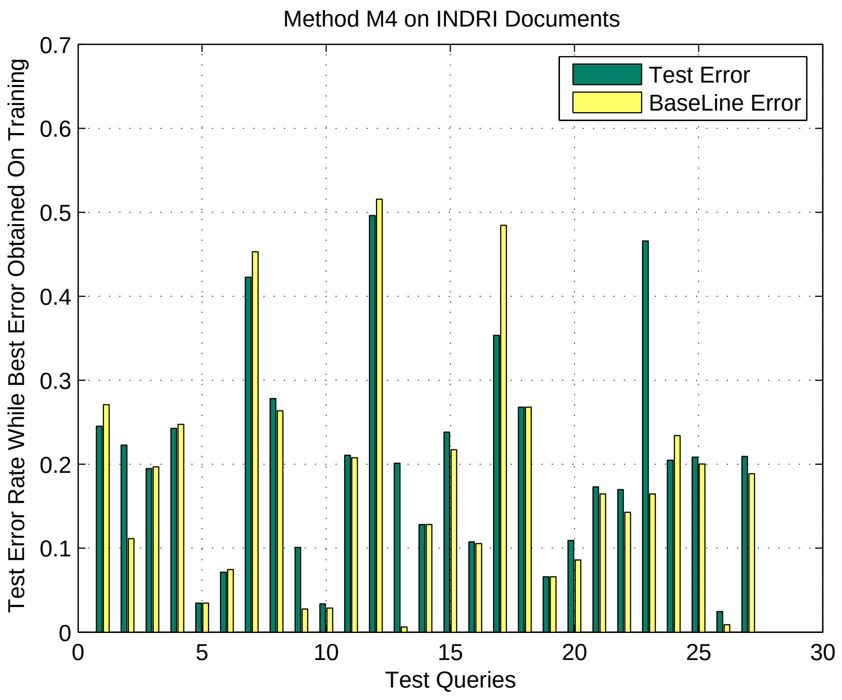
Table 6.23: Average classifier performance versus average baseline performance, while having the best results for training set in method **M4** on **INDRI** documents corresponding to GALE-Y1 queries.

	Improvement	**No Changes**	**Degradation**
Error rate	12	0	15
F-measure	10	0	17

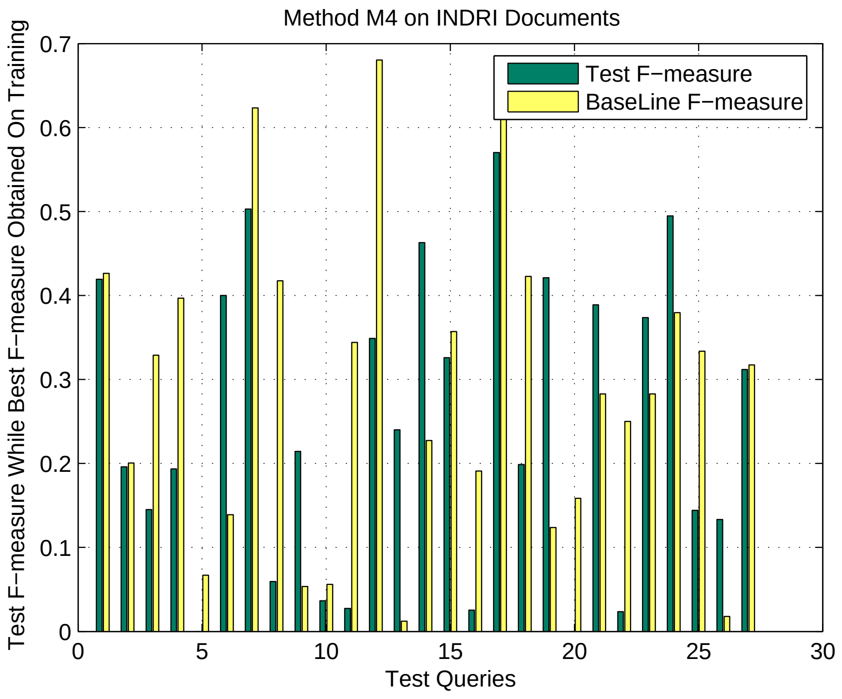
Table 6.24: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M5** on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M5	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.1937	0.1814	-6.7669	N/A
F-measure	0.2553	0.2867	-10.9501	N/A

Table 6.25: Average classifier performance versus average baseline performance, while having the best results for training set in method **M5** on **INDRI** documents corresponding to GALE-Y1 queries.

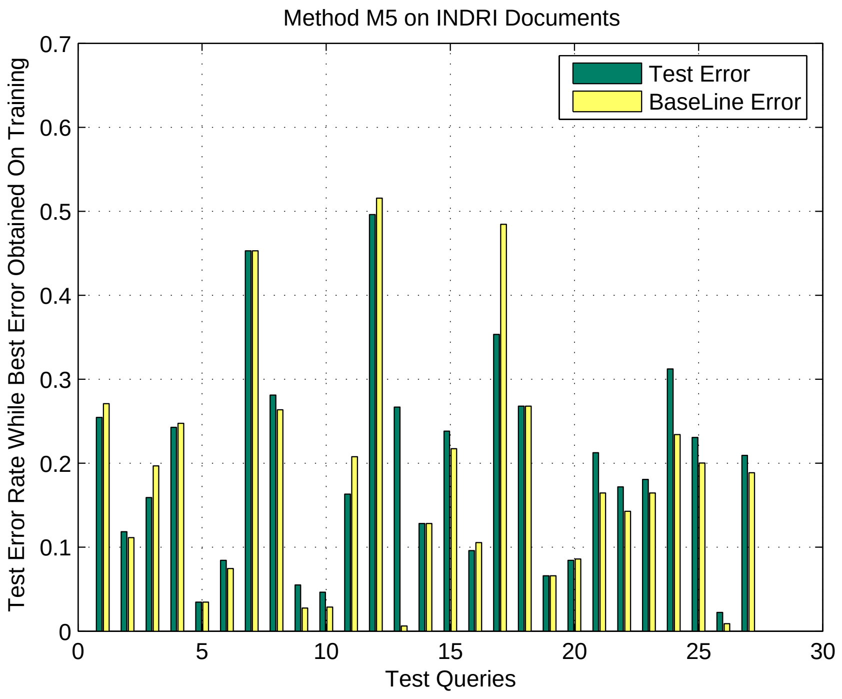


(a) Test error rate of Method M4 Under Best Results For Training Set

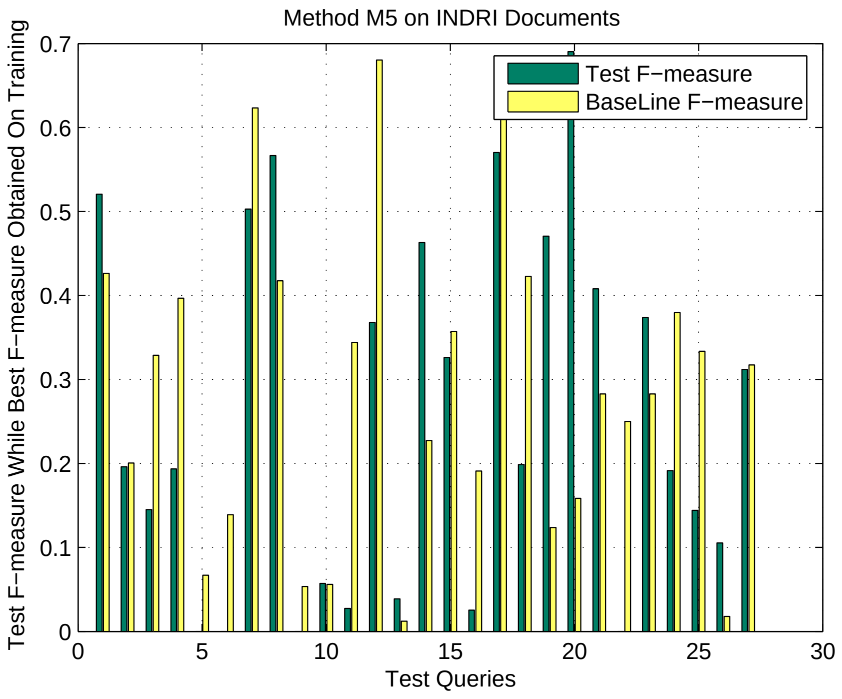


(b) Test F-measure of Method M4 Under Best Results For Training Set

FIG. 6.13: **Test error rate** and **F-measure** while having best training classification error when method **M4** is executed on **INDRI** documents corresponding to GALE-Y1 queries.



(a) Test error rate of Method M5 Under Best Results For Training Set



(b) Test F-measure of Method M5 Under Best Results For Training Set

FIG. 6.14: **Test error rate** and **F-measure** while having best training classification error when method **M5** is executed on **INDRI** documents corresponding to GALE-Y1 queries.

M6: All-Words Selection

The result sets can be observed in figure 6.15. Tables 6.26and 6.27 show the numeric comparison of error or F-measure changes, between best classification result and baseline while having best training results.

	Improvement	No Changes	Degradation
Error rate	8	0	19
F-measure	4	0	23

Table 6.26: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M6** on **INDRI** documents corresponding to GALE-Y1 queries.Test and training queries come from **GALE-Y1** source.

	Method M6	Baseline	Relative Improvement %	Stat. Sig. Test Level
Error rate	0.1844	0.1814	-1.6299	N/A
F-measure	0.1619	0.2867	-43.5265	N/A

Table 6.27: Average classifier performance versus average baseline performance, while having the best results for training set in method **M6** on **INDRI** documents corresponding to GALE-Y1 queries.Test and training queries come from **GALE-Y1** source.

Similar to the results in the previous section, this method resulted in huge performance drop for the F-measure. and little degradation for the classification error rate. This shows the necessity of an alternative method for improving of F-measure.

M7: Loose Word Selection

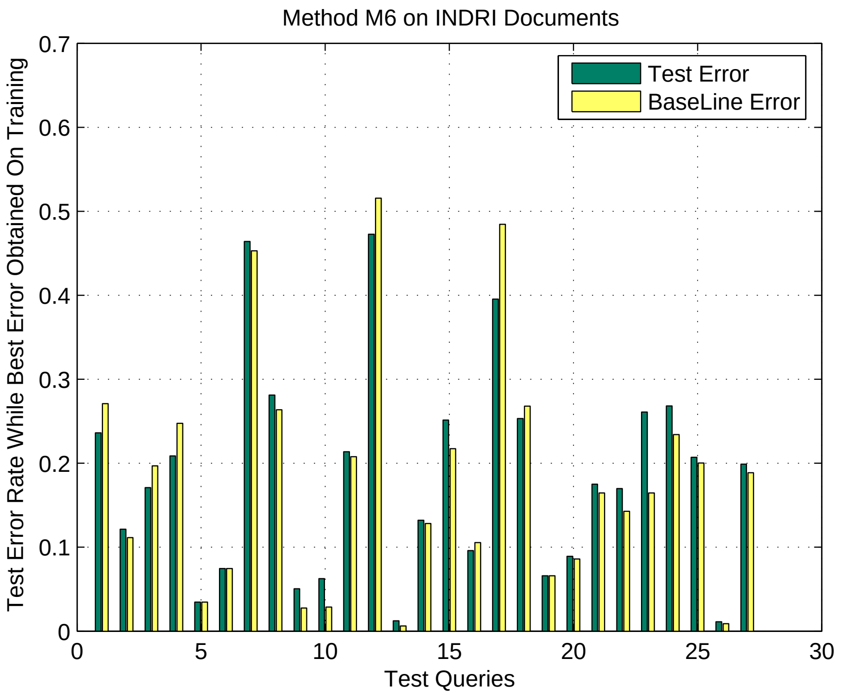
	Improvement	No Changes	Degradation
Error rate	14	0	13
F-measure	22	0	5

Table 6.28: This table shows number of cases that improvement or no change or degradation of classification error or F-measure has happened comparing to baseline, in method **M7** on **INDRI** documents corresponding to GALE-Y1 queries.

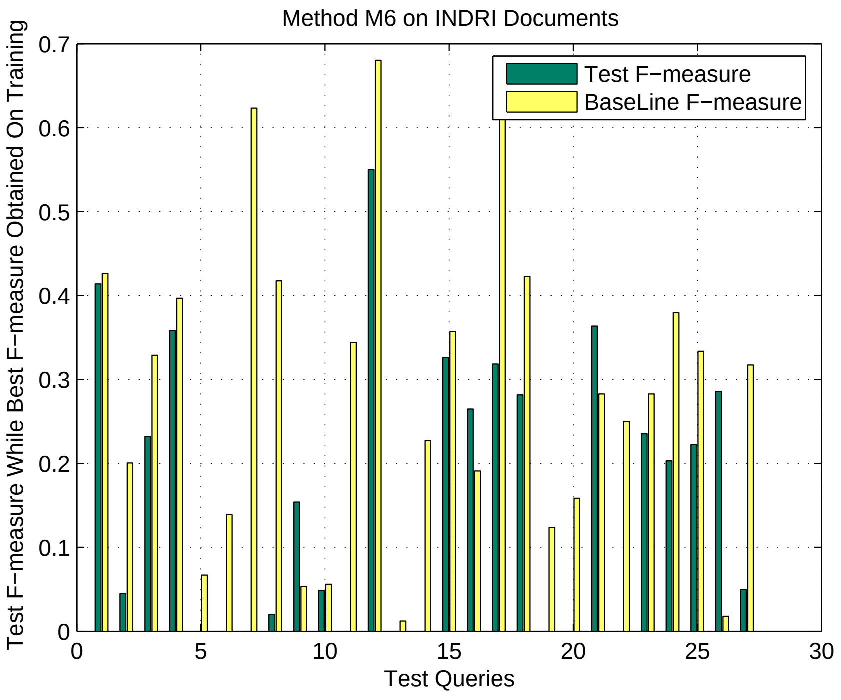
The results show improvements both on CER and F-measure. The best improvement on F-measure is obtained by this method up to now and the results is correct with probabilty of 99.9%. Since INDRI data is much similar to real data, it shows that this method is the best method among all for F-measure improvement, but the result on CER is not that significant. F-measure is a very important parameter for this kind of text processing tasks, therefore we can apply this method in same kind of applications.

6.3. Summary of Reslts

Tables 6.30, 6.31, 6.32 and 6.33 summarize the results presented so far on GALE-Y1 corpus. In the same way, tables 6.34, 6.35, 6.36 and 6.37 show a summary of results when the training queries are extracted from GALE-Y1 corpus and test queries from GALE-Y2.

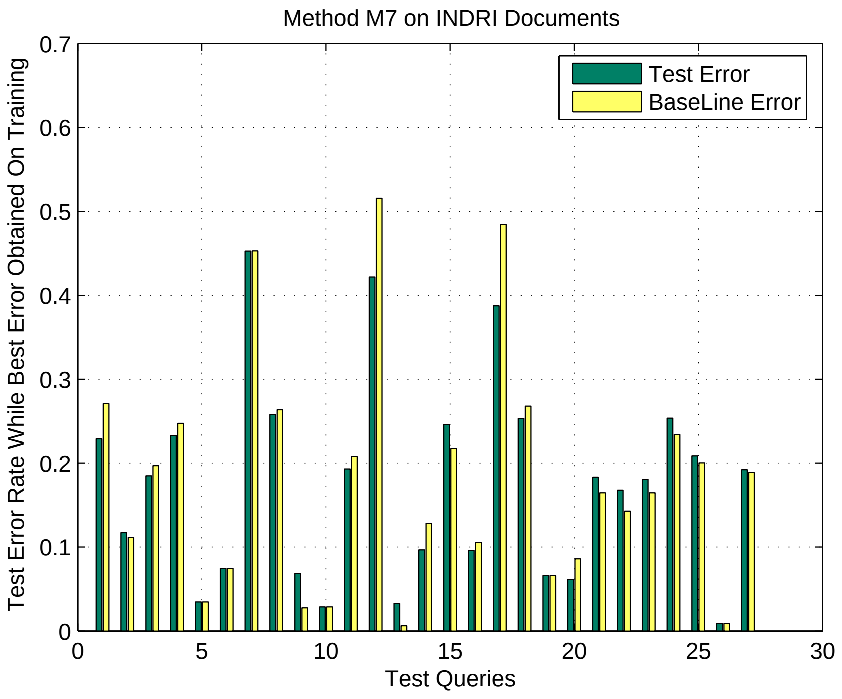


(a) Test error rate of Method M6 Under Best Results For Training Set

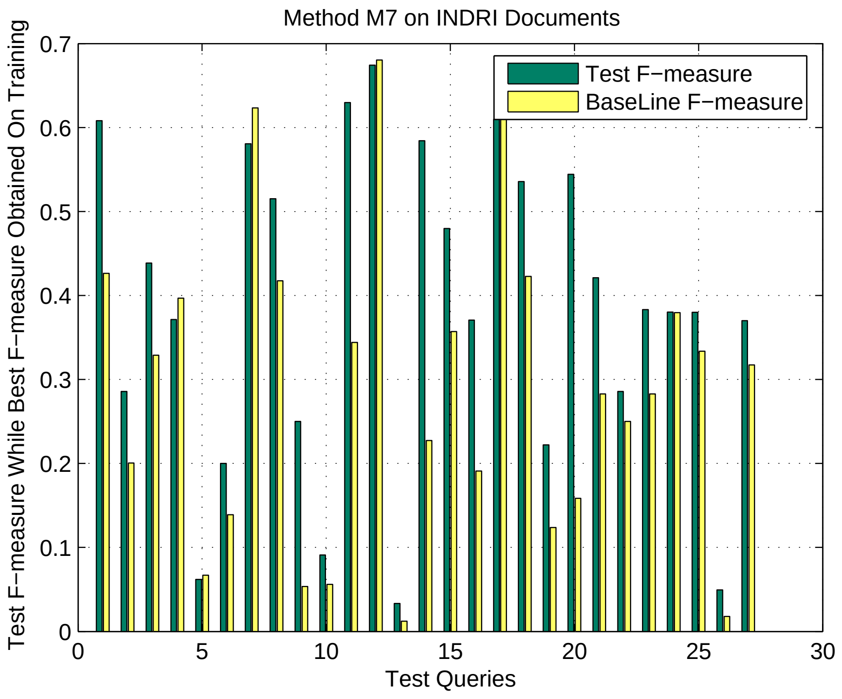


(b) Test F-measure of Method M6 Under Best Results For Training Set

FIG. 6.15: **Test error rate** and **F-measure** while having best training classification error when method **M6** is executed on **INDRI** documents. Training queries come from **GALE-Y1**.



(a) Test error rate of Method M7 Under Best Results For Training Set



(b) Test F-measure of Method M7 Under Best Results For Training Set

FIG. 6.16: **Test error rate** and **F-measure** while having best training classification error when method **M7** is executed on **INDRI** documents corresponding to GALE-Y1 queries.

	Method M2	**Baseline**	**Relative Improvement %**	**Stat. Sig. Test Level**
Error rate	0.1752	0.1814	**3.409**	–
F-measure	0.3836	0.2867	**33.7615**	**0.001**

Table 6.29: Average classifier performance versus average baseline performance, while having the best results for training set in method **M7** on **INDRI** documents corresponding to GALE-Y1 queries.

As seen, each method has strengths and weaknesses. The performance varies depending on the corpus. In some of the methods, CER decreases while F-measure also decreases. In those cases, the method is capable of extracting correct irrelevant sentences, but it detects more than necessary. Since the majority of sentences are irrelevant, it improves the CER, but hurts the F-measure.

For GALE-Y1 queries, we obtained the best F-measure and CER improvement by method M7 when using LDC and INDRI data. On LDC data, also M5 and M6 work good for CER reduction.

Similarly we obtained F-measure improvements using M2 and M3. It means that less noisy data, helps to increase F-measure, although does not necessarily reduce the CER. In other words, it means that even if we lose some data, the selected sentences are classified correctly with high confidence.

	Test Source	**LDC**	**INDRI**
M1	Y1	.38	-11.25
M2	Y1	-.65	-16.15
M3	Y1	-.65	-16.15
M4	Y1	-2.67	-11.91
M5	Y1	3.63	-6.7
M6	Y1	5.30	-1.6
M7	Y1	6.63	3.40

Table 6.30: This table shows a comparison between the performance of different methods in terms of **classification error rate** over LDC and INDRI data for **GALE-Y1** queries. The results show the percentage of relative improvement, compared to baseline.

	Test Source	**LDC**	**INDRI**
M1	Y1	.77	-3.44
M2	Y1	3.83	11.46
M3	Y1	6.55	3.61
M4	Y1	-27.33	-14
M5	Y1	-21.03	-10.95
M6	Y1	-52.08	-43.52
M7	Y1	18.79	33.76

Table 6.31: This table shows a comparison between the performance of different methods in terms of **F-measure** over LDC data and INDRI data for **GALE-Y1** queries. The results show the percentage of relative improvement, compared to baseline.

For GALE-Y2 queries, almost all of the methods have some improvements of CER on LDC data. LDC data is clean data which contains at least some relevant sentences. Methods, M1, M4, M5 , M6 and M7 perform the same even on INDRI data. It shows that the choice of using less noisy data or TF-IDF or other criteria, depends strongly on the corpus. In the case of F-measure, as it was expected, M7 has the best performance and much better than other methods.

	LDC	**INDRI**
Best Method	M7	M7
Best Error Rate Reduction	6.63	3.40

Table 6.32: The methods having **best performance on Classification Error** on LDC and INDRI data respectively, are listed in this table.

	LDC	**INDRI**
Best Method	M7	M7
Best F-measure improvement	18.79	33.76

Table 6.33: The methods which perform the **best** in terms of **F-measure** on LDC and INDRI data respectively, are listed in this table.

	Test Source	**LDC**	**INDRI**
M1	Y2	17.66	5.86
M2	Y2	14.28	-7.62
M3	Y2	14.28	-7.62
M4	Y2	2.40	7.83
M5	Y2	.91	5.08
M6	Y2	11.36	7.78
M7	Y2	3.46	7.00

Table 6.34: This table shows a comparison between the performance of different methods in terms of **Classification Error Rate** over LDC data and INDRI data for **GALE-Y2** queries. The results show the percentage of relative improvement, compared to baseline.

	Test Source	**LDC**	**INDRI**
M1	Y2	-11.76	-10.36
M2	Y2	-10.53	-17.39
M3	Y2	-10.53	-8.35
M4	Y2	-61.92	-23.34
M5	Y2	-38.66	-22.87
M6	Y2	-61.07	-55.52
M7	Y2	30.82	28.82

Table 6.35: This table shows a comparison between the performance of different methods in terms of **F-measure** over LDC and INDRI data for **GALE-Y2** queries. The results show the percentage of relative improvement, compared to baseline.

	LDC	**INDRI**
Best Method	M1	M4
Best Error Rate Reduction	17.66	7.83

Table 6.36: The methods having **best performance on Classification Error** on LDC and INDRI data respectively for **GALE-Y2** queries, are listed in this table.

	LDC	**INDRI**
Best Method	M7	M7
Best F-measure improvement	30.82	28.82

Table 6.37: The methods which perform the **best** in terms of **F-measure** on LDC and INDRI data respectively for **GALE-Y2** queries, are listed in this table.

CHAPTER
7

Conclusions

Template 1 queries are very general questions with a wide domain of responses. Although using a template, the question answering is quite different from that of other templates, since all the information about the event is concentrated in the slot. Answering to these kind of questions is much similar to extraction of general passages from an IR engine. If the performance of classification for this template is increased, it would be useful to answer free-template questions as well.

In supervised classification methods, sufficiently large size of annotated data is required to train a system. However, annotated data may not be available for particular query and it is time consuming to label them manually. Therefore, we were motivated to implement an efficient unsupervised distillation method to automate sentence annotating in response to template 1 queries and reduce labeling efforts in the framework of GALE Project. Later, other templates may also exploit similar unsupervised methods.

Throughout this work, we used the documents returned either by the manual annotations provided by LDC or by the University of Massachusetts INDRI IR engine. The approach consists of using only highly confident features such as word transcriptions extracted from query as well as their synonyms. After forming the bootstrap model using these features, the model is improved using self-training, and is iteratively trained during consecutive runs.

In various experiments, we use statistical sentence extraction for information distillation. We observed that less noisy data helps to find a clean collection of relevant and irrelevant sentences. But in the meanwhile, depending on the word filtering (either by means of word frequencies or by TF-IDF), risk of losing a large amount of sentences in training data increases. The methods which use this kind of selection have deficiencies in terms of error rate enhancement, since the collection of non-labeled sentences should be labeled randomly by classifier. However, in some of them F-measure is improved significantly. It should also be considered that we have a tough baseline for F-measure which its recall is 100%. Thus, beating such a baseline is not a simple task.

This approach tries to enhance the F-measure by increasing recall and in the meanwhile uses confident features to find relevant sentences and consequently increase the error rate. Therefore, it seems to output more coherent results comparing to previous attempts.

Future Work

In this work, we tested only English documents and sentences of our corpus. However, as a novel work, unsupervised distillation can be applied into other languages translated in English. It can be applied and tested to perform natural language information distillation, as well.

The improvement of classification is related to extraction of proper features. Another important challenge would be to use other sophisticate features rather than just word transcriptions and synonyms, including named entities, co-references (mentions), relationships and events in the document sources. These are IE elements and preferred to be extracted from ACE (Automatic Content Extraction) annotation guidelines, defined by the annual NIST ACE evaluations.

It would even be more interesting to try compare the performance of the methods presented in this work with other baselines rather than chance baselines. As an example it could be to use *Monte Carlo Baseline*. Monte Carlo search refers to the extreme of simply randomly choosing states and keeping track of the best so far. To have a fair baseline, we can assign x% of the sentences as relevant in each document and then compute F-measure and error rate. X% is the prior probability. For example if 10% of the sentences are relevant, assign one out of 10 sentences as relevant randomly.

Bibliography

BLOEHDORN, S., & HOTHO, A. 2005. Text Classification by Boosting Weak Learners based on Terms and Concepts. 8

BLUM, A., & LANLGEY, P. 1997. Selection of Relevant Features and Examples in Machine Learning. *Artificial Intelligence*.

BREIMAN, L. 1996. *Bias, variance and arcing classifiers*. Tech. rept. University of California Berkeley.

CHAPPELIER, J.C., & RAJMAN, M. 2006a. *Information Extraction Out Of Textual Data*.

CHAPPELIER, J.C., & RAJMAN, M. 2006b. *NLP Evaluation and Linguistic Resources*.

DODDINGTON, G., MITCHELL, A., PRZYBOCKI, M., RAMSHAW, L., STRASSEL, S., & WEISCHEDEL, R. 2004. The Automatic Content Extraction (ACE) Program Tasks, Data, and Evaluation. 19

FAZLY, A., NORTH, R., & STEVENSON, S. 2006. Automatically Determinable Allowable Combinations of a Class of Flexible Multiword Expressions. 38

FELLBAUM, CH. 1998. *Wordnet: An Electronic Lexical Database*. Bradford Books. 27

FLEURET, F. 2006. *Short Note: Boosting as a Gradient Descent*. Demonstration of Boosting being Gradient Descent algorithm.

FREITAG, D. 2000. Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, **39**(2-3), 169–202. 5

FREUND, Y., & SCHAPIRE, R. 1995. A decision-theoretic generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*. 9

FREUND, Y., & SCHAPIRE, R. 1999. A Short Introduction to Boosting. *Japanese Society for Artificial Intelligence*. 10

FRIEDMAN, J., HASTIE, T., & TIBSHIRANI, R. 1998. *Additive logistic regression: a statistical view of boosting*. 9

GALE.

GORGEL, P., & O.N.UKAN. 2007. Performance Analysis of Neural Network Handwritten Character Recognition System Using CNN EDGE Detection. 35

GROUP, NIST. *Introduction to Information Extraction*.

HAKKANI-TÜR, D., & TUR, GOKHAN. 2007. Statistical Sentence Extraction For Information Distillation. 1, 13, 20

HAKKANI-TÜR, D., TUR, G., & LEVIT, M. 2007. Exploiting Information Extraction Annotations for Document Retrieval in Distillation Tasks. 1, 2, 19, 34

HAND, D.J., MANNILA, H., & SMYTH, P. 2001. *Principles of Data Mining*. 39

KEARNS, M. 1998. *Thoughts on hypothesis boosting*. 8

LEVIT, M., HAKKANI-TÜR, D., TUR, G., & GILLICK, D. 2007a. Integrating Several Annotation Layers For Statistical Information Distillation. 2, 21, 24

LEVIT, M., BOSCHEE, E., & FREEDMAN, M. 2007b. Selecting On-Topic Sentences from Natural Language Corpora. 19

LEWIS, DAVID D. 1992. *Feature Selection and Feature Extraction for Text Categorization*.

LIU, Y., SHRIBERG, E., STOLCKE, A., HILLARD, D., OSTENDORF, M., PESKIN, B., & HARPER, M. 2006. The ICSI-SRI Spring 2006 Meeting Recognition System. *In: Machine Learning for Multimodal Interaction: Third International Workshop*. 8

MCCALLUM, A. 2005. *Information extraction: distilling structured data from unstructured text*. Vol. 3. ACM Press. Pages 48–57.

REYZIN, L., & SCHAPIRE, R. 2006. How boosting the margin can also boost classifier complexity. *In: Proceedings of the 23rd Internation Conference on Machine Learning*.

SCHAPIRE, R. 2001. The Boosting Approach to Machine Learning: An Overview. *In: Proceedings of MSRI Workshop on Nonlinear Estimation and Classification*.

SCHAPIRE, R., & SINGER, Y. 2000. Boostexter: A Boosting-based System for Text Categorization. *Machine Learning*, **39**(2/3). 8

SCHAPIRE, R., FREUND, Y., BARTLETT, P., & LEE, W. 1998. Boosting the margin: a new explanation for the effectiveness of voting methods. *The Annals of Statistics*.

SCHAPIRE, R., ROCHERY, M., RAHIM, M., & GUPTA, N. 2005. Boosting with prior knowledge for call classification. *In: IEEE Transactions on Speech and Audio Processing*.

SCHIFFMAN, B., & MCKEOWN, K. R. 2007. Question Answering using Integrated Information Retrieval and Information Extraction. 21

SEQUEL. 2007. *Statistical Leanring*. 5

SHAVER, J.P. 1993. What Statistical Significance Testing Is, and What It is Not. Journal of Experimental Education. 39

STENCHIKOVA, SS, HAKKANI-TÜR, D., & TUR, G. 2006. QASR: Question Answering Using Semantic Roles for Speech Interface. *INTERSPEECH 2006*.

STROHMAN, T., METZLER, D., TURTLE, H., & CROFT, W. Indri: A language-model based search engine for complex queries. 20

VOORHEES, E., & TICE, D. 2000. Building a Question Answering Test Collection, , July, 2000, pp. 200-207. *Proceedings of SIGIR-2000*.

VOORHEES, E. M. 2003. Overview of the trec 2003 question answering track. 1

WAYNE, CH. June 1998. Topic Detection and Tracking (TDT) Overview and Perspective. *In: Proceedings of DARPA Broadcast News Tracsription and Understanding Workshop*. 33

ZIMMERMANN, M., HAKKANI-TŘ, D., FUNG, J., MIRGHAFORI, N., E.SHRIBERG, & LIU, Y. 2006. The ICSI+ Multi-Lingual Sentence Segmentation System. *In: Proceedings of ICSLP*.