PRESENTATION
Overview of Personalized Information Retrieval Track @ FIRE-2011

Debasis Ganguly, Johannes Leveling, Keith Curtis, Wei Li, Gareth Jones
School of Computing, CNGL, Dublin City University

Outline

- TREC-style evaluation
- Beyond TREC: Personalized IR task objectives
- Data preparation
- Information retrieval with logs
- Results
- Conclusions and future work
TREC-style Evaluation

- Information Retrieval evaluation campaigns
  - TREC, CLEF, NTCIR, INEX, FIRE
- Evaluation methodology:
  - Organizer(s):
    create topics (queries)
  - Participating systems:
    index document collection, process topics, submit results
  - Organiser(s) (+ participants):
    evaluate submissions

Example TREC topic

<top>
<num>401</num>
<title>foreign minorities, Germany</title>
desc>What language and cultural differences impede the integration of foreign minorities in Germany?</desc>
narr>A relevant document will focus on the causes of the lack of integration in a significant way; that is, the mere mention of immigration difficulties is not relevant. Documents that discuss immigration problems unrelated to Germany are also not relevant.</narr>
</top>
PIR Task Motivation (beyond TREC)

- Limitations of ad-hoc IR evaluation paradigm:
  - One topic (query) fits all (users)
  - One result set fits all (users)

- PIR task: Log the topic development to enable research in personalisation
  - Different users have different ways of expressing the same information need.
  - Different users formulate topics for same broad search categories (e.g. "Bollywood movies")
  - Users are not really sure what they are looking for initially. So querying is an iterative process e.g. "education in India" -> "top engineering institutes of India" -> "research in IIT Bombay" etc.

PIR Task Motivation (contd.)

- Elements of personalization:
  - Different query formulations and relevant documents
  - Same user develops topic and judges relevance
  - Topic development and evaluation on same corpus
  - (→ reproducible results)

- Elements of collaboration:
  - Users choose a search category as their starting points.
  - Two users with same category indicate users with similar interests.

- Research Question:
  - Can we tune IR systems to address individual user-specific information needs?
Difference with ad-hoc topics

TREC: Q₁ -> D₁, ..., Dₘ
(same query, single result set)

Category → Q₁ → Q₂ → ... → Qₙ
D₁¹ D₂¹ ... Dₘ¹
D₁² D₂² ... Dₘ²
...                ...
D₁ⁿ⁻¹ D₂ⁿ⁻¹ ... Dₘⁿ⁻¹
→ Qₙ

PIR track activity flow
Data Preparation (1/3)

- Document collection –
  - English FIRE-2011 ad-hoc collection (articles from Indian and Bangladesh newspapers)
  - Index the collection with Lucene
  - Identify 15 broad category news domains

- Java Servlet based search interface which supports
  - user registration to maintain user identity
  - category selection and navigation
  - document retrieval against a user query
  - viewing and bookmarking documents
  - submission of summary and final test topic.

Data Preparation (2/3)

- Each line in a CSV formatted log contains:
  - User name (U), category name (C) – to identify the current session.
  - Query string (Q) – to identify the current query.
  - Document ID (D) – to identify the current document on which an action is performed
  - Action – click to view or bookmark
  - Timestamp – to compute relative viewing times
Data Preparation (3/3)

- Queries in extended TREC format (final test topics)
- Additional fields:
  - User name of the topic developer
  - Topic category

Information Retrieval using Logs

- Question:
  - Can we tune IR systems to address individual user-specific information needs?

- Objective:
  - Investigate benefit of using additional information about a user (topic developer) on IR performance

- Data:
  - Ad hoc document collection
  - User information (search history+search categories)
  - Final topics

- Evaluation metrics:
  - P@5, P@10, (MAP)
Not everything goes according to plan 😞

- 26 registered participants, but no run submissions!
- 25 topics: enough for standard IR evaluation, but not enough for PIR
- 10 topic developers with different interests
- Very small overlap
  - Categories:
    - Indian tourism – 5
    - Relation of India with its neighbouring countries – 4
    - Indian traditions and customs – 3
    - Indian paintings and painters – 3
  - Documents: Virtually no overlap – at most 2

Never lose hope 😊

- We generated three baseline submissions:
  - BL1: standard IR run on test topic titles (no information from logs used!)
    - Pseudo-relevance Feedback (PRF) with R=10 and T=10
  - BL2: standard IR run using intermediate query titles of a given test topic as a query
    - PRF with R=10 and T=10
  - BL3: standard IR run using viewed documents as additional pseudo-relevant documents
    - PRF with R>= 10 and T=10

Document pool constructed from baseline runs
A data structure for efficient log processing

- Two level hash table with list of list
- Quickly retrieves the intermediate queries and viewed docs for a given category and user name

Results

<table>
<thead>
<tr>
<th>Run Name</th>
<th>P@5</th>
<th>P@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>BL1</td>
<td>0.368</td>
<td>0.356</td>
</tr>
<tr>
<td>BL2</td>
<td>0.400</td>
<td>0.352</td>
</tr>
<tr>
<td>BL3</td>
<td>0.384</td>
<td>0.300</td>
</tr>
</tbody>
</table>

A sample instance:

Topic number 1 “rock climbing india”
- BL1: Ranks of relevant docs = \{27,184,303\}
- BL2: Ranks of relevant docs = \{2,3,4\}
- BL3: Ranks of relevant docs = \{1,2,4\}
Conclusions and Future Work

- Results show that logs can be used to improve precision at top ranks

- If two simple approaches work reasonably well, then more complex methods may work even better. For example
  - Using the view times to predict relevance
  - Using the bookmarks as pseudo-relevant documents
  - Using RS techniques such as popularity of a document across users

Future Work

- Identify reasons for lack of submissions

- Generate more logs from more users

- Make the search task more interesting by using Wikipedia instead of news
Please provide your suggestions and comments and express your interest for a PIR task 2012

THANK YOU... 😊
Cross Lingual Text Reuse Detection Using Key phrase Extraction and Similarity Measures

by Rambhoopal K. Vasudeva Varma

Overview

- Introduction
- About CL!TR
- Related work
- Approaches
- Experimental Evaluation
- Conclusions and Future Work
Introduction

- What is Text Reuse?
  - Text reuse is an imitate of phrases from others text documents and presenting them as their own.

- How hard is the problem?
  - Reused text is commonly modified aiming to hide the plagiarism.
  - Not all similar text sections are examples of Text Reuse.

Introduction

- Text reuse detection across languages is even more harder.
- Detection of text reuse between distant language pairs is most difficult of all.
Related work

- Most approaches have addressed text reuse issue by measuring
  - Lexical and structural similarity of documents:
    - Individual words.
    - Fixed length substrings (e.g. n-grams).
    - Variable-length substring.

Related work

- Drawbacks of these approaches:
  - Their decision/classification based on one single feature such as degree of overlap.
  - Common domain-specific word sequences causes an overestimation of their overlap
About CL!TR

- CL!TR (Cross language Indian Text Reuse) task is one variant of FIRE 2011 (Forum for Information Retrieval Evaluation).
- CL!TR task is to identify the text reuse documents given suspicious documents in Hindi and source document in English.
- We have used CL!TR task data collection for all our experiments to detect text reused documents across languages.

Approaches

- We implemented three approaches in text reuse detection
  - First: n-grams and cosine similarity.
  - Second: Features denoting common text relevance and fragmentation.
  - Third: Automatic extraction of Key phrases and new measure for similarity while using an open source search engine.
First Approach

- Tri-grams extracted using sliding window for both suspicious and source documents.
- Cosine Similarity between tri-grams of each suspicious document against with all tri-grams of all source documents.
- Top similarity scored source document as text reuse source for creating suspicious document.
- J48 Decision tree classifier using WEKA tool with cosine similarity score as single feature for classification of documents

Second Approach

- Set of features denoting common text fragmentation and relevance.
- Fragmentation feature is based on two basic assumptions
  - Longer the sequence the greater the evidence of text reuse.
  - More common the sequence, greater the evidence of text reuse (since long sequence are very rare)
Second Approach

- Relevance Feature is based on a measure of relevance of frequent sequences.
- This measure of relevance has two parts
  - We evaluate how frequent is the given sequence in the suspicious document to penalize frequent sequences because of their probability of being domain specific.
  - Castigates the sequence formed by words that are frequent in both documents.

Third Approach

- Use keyphrase instead of n-grams
  - Keyphrase are sequence of words that captures the main topics in the document.
- For automatic keyphrase extraction we use following techniques
  - n-gram filtration technique.
  - term weighting technique.
Third Approach

- n-gram filtration technique
  - use LZ78 data compression techniques with simple refinement steps.
  - use pattern filtration techniques.

- term weighting technique
  - use position of phrase in the sentence.
  - use position of sentence of the given phrase in the document.

Third Approach

- Use an open source search engine *Nutch* for similarity measure.

- Index all source documents, use Opic-scoring algorithm to calculate the document score.
Third Approach

- Source documents as index and keyphrases of given suspicious document as queries we retrieve all the relevant source documents.
- We get several groups of retrieved source documents for all keyphrases of given suspicious document.
- We create a list of unique source documents with their frequency score by combining all the groups of retrieved source documents for all keyphrases of a suspicious document.

Third Approach

- For each suspicious document we create this list of unique source documents with frequency score.
- The highest frequent score source document from the list of a suspicious document is considered source of its text reuse.
- We use minimum threshold of 31(frequency score) to consider a document as source for text reuse.
- Threshold is based on the development corpus.
Experimental Evaluation: Corpus

- We used CL!TR data collection for all our experiments.
- Training data contains 198 suspicious documents in Hindi and 5032 source documents in English.
- Testing data contains 190 suspicious documents in Hindi and 5032 source document in English.
- Training data of 198 suspicious documents is as follows:
  - 130 are given as positive examples of text reuse documents.
  - 68 are given as negative examples of text reuse documents.

Experimental Evaluation

- We used Google translator for translating all suspicious documents in training and testing data.
- We translated Hindi suspicious documents to English.
- We used English stop word list in all our approaches to remove them from training and testing documents while pre-processing the documents and contains 173 entries.
- We used English Porter stemmer for first and second approaches.
Experimental Evaluation

- **Evaluation**
  - CL!TR testing data given 190 testing suspicious documents following are the number of text reused documents considered by our approaches.
    - First approach considered 117 suspicious documents as text reused.
    - Second approach considered 125 suspicious documents as text reused.
    - Third approach considered 147 suspicious documents as text reused.

- There were total of 15 runs submitted for CL!TR task in FIRE-2011.

- Following are the results of our approaches
  - Our third approach secured first rank with F-measure of 0.649 in CL!TR 2011 task.
  - Second approach secured third rank with F-measure of 0.608.
  - First approach secured 7th place.
Conclusion and Future work

- Differentiator: Usage of keyphrases instead of n-grams and a new measure for similarity for text reuse detection.
- Features denoting relevance and fragmentation of common sequences was also good but with less/poor recall.
- Future work: Explore how semantic text feature applied across languages between distant language pairs could improve the F-measure.

Thank You
SMS based FAQ retrieval

Nishit Shivhare
RVCE, Bangalore

CONTENTS

Introduction
Problem Statement
System Implementation
Experiments and Results
Conclusion and Future Work
INTRODUCTION

This FAQ Retrieval system is designed to find a match from the given set of Frequently Asked Questions for a query written in SMS language. The problem with questions asked in SMS language is that the SMS text has a lot of noise. Understanding user questions in natural languages requires Natural Language Processing (NLP). The match $Q^*$ for the SMS query is found out by using sequence matching techniques, disemvoweling, etc.

Problem Statement

In this task, we have a corpus of frequently asked questions and answers from various domains that have been provided. The corpora of questions in the database are represented by $Q$. The query is in SMS language which may or may not contain noise. The goal of the task is to find a question $Q^*$ from the corpora of FAQ’s $Q$, that is the best possible match for the SMS query $S$. 
Problem statement contd..

I have two parameters for calculating the score of a question, keyword score and similarity score.
The methods for calculating the keyword score, like disemvoweling, are based on the general observations made about the language and slangs used by people while typing SMS text.
On the other hand, the similarity score is calculated using dynamic programming techniques for string comparison and pattern matching algorithms, like Longest Common Subsequence and Gestalt Pattern Matching.

System Implementation

- Preprocessing
- Disemvoweling
- Removal of stop words
- Keyword matching

- Calculation of weight of each word using:
  * Similarity ratio
  * Longest Common Subsequence ratio
  * Levenshtein Distance
  * Inverse Document Frequency

- Creation of variant lists for each SMS word
- Similarity score
- Total Score

Rishit Shihore, RVCE
Preprocessing

- We create a hash table of words $W$ that contains all the words occurring in all the questions in $Q$ with the keys being characters a-z and numbers 0-9.
- Example: ‘i’ contains all the words in the set $Q$ that start with ‘i’, like ‘insurance’, ‘improve’, and so on.
- A list of stop words is also prepared and disemvoweled.
- Digits occurring in SMS token are replaced by a string based on a manually designed digit-to-string mapping (“8”->“eight”).
- Single character words in the SMS query are removed.

Disemvoweling

- We describe the process of removing vowels from a string as disemvoweling and the string from which vowels are removed is said to be disemvoweled.
- We apply this process of disemvoweling to the SMS query because in general, it has been observed that the user tries to compress the text by removing vowels.
Removal of stop words

- Stop words are words which are filtered out prior to, or after, processing text.
- The stop words are removed from the SMS query S. We now call it processed SMS query.
- The list of stop words that we have used includes the most common short function words such as \textit{the}, \textit{is}, \textit{at}, \textit{which}, \textit{on}, etc. and common lexical words as well.

Keyword Matching

- In order to calculate the \textit{keyword score} of a question $q$ in $Q$-Poss (described later), we disemvowel all the questions and for each question $q$ we find the number of words of the SMS query it contains.
- Words or tokens of the query are called keywords.

\text{Keywords score}(q) = \frac{\text{No. of keywords in } q}{\text{No. of keywords in } S} \quad (1)
Calculation of weight of a word

- For each token of the SMS query (not disemvoweled), we calculate its similarity with every word w in the corpus W. The weight of a word is given by the equation:

\[
\text{Weight}(w, s) = \frac{\text{LCSR}(w, s) \times \text{SMRatio}(w, s) \times \text{IDF}(w)}{\text{LevDistance}(w, s)} \quad \text{(2)}
\]

* \( \text{LCSR}(w, s) \) - Longest Common Subsequence Ratio of the SMS query token s and the word w in W.  
* \( \text{SMRatio}(w, s) \) - Similarity ratio using Ratcliff/Obershelp algorithm.  
* \( \text{LevDistance}(w, s) \) - Levenshtein Distance between disemvoweled w and s  
* \( \text{LevDistance}(w, s) \) - Levenshtein Distance between disemvoweled w and s

Longest Common Subsequence Ratio (LCSR)

- The longest common subsequence (LCS) problem is to find the longest subsequence common to all the sequences, in our case LCS of w and s. The LCS ratio is given by:

\[
\text{LCSR}(w, s) = \frac{\text{LCS}(w, s)}{\max(\text{length}(w, s))} \quad \text{(3)}
\]
Similarity ratio

- The similarity ratio (SMRatio) of two words is calculated by an algorithm that predates the Ratcliff/Obershale algorithm for 'gestalt pattern matching'.
- The algorithm works by examining two strings passed to it and locating the largest group of characters in common. The algorithm uses this group of characters as an anchor between the two strings. The algorithm then places any group of characters found to the left or the right of this anchor on a stack for further examination.

\[
\text{SMRatio}(w,s) = \frac{2 \times \text{No. of common characters}}{\text{No. of characters in the two strings}} \quad \ldots(4)
\]

Levenshtein Distance

- Levenshtein distance is a 'distance' between two strings given by counting the minimum number of operations needed to transform one string into the other, where an operation is defined as an insertion, deletion, or substitution of a single character, or a transposition of two characters.
- w and s are disemvoweled to calculate the Levenshtein distance between them.
Inverse Document Frequency (IDF)

- If $f$ number of documents in corpus $Q$ contain a term $w$ and the total number of documents in $Q$ is $N$, the Inverse Document Frequency (IDF) of word $w$ is
  \[
  \text{IDF}(w) = \log \frac{N}{f} \quad \ldots(5)
  \]
- A word which occurs less number of times in the corpus $Q$ will have a high IDF.
- The reason behind this logic is that queries are composed of informative words.

Creation of variant lists

In order to calculate the similarity score of each question in $Q$, we first create a variant list for each SMS token. This is done by calculating weight of each word $w$ in $W$ with respect to each SMS token $s$ using equation (2). This list is then sorted in descending order. The list contains the top 5 variants. A word is said to be a variant of the SMS token if it starts with the same character and if \[\text{Length (LCS (w, s))} > 1\]
Variant lists contd..

A search is performed on the corpus $Q$ for the questions that contain the variant $w$ and all these question are added to a list called $Q$-Poss. Thus, after the search for variants for the SMS tokens is complete, $Q$-POSS will contain all the questions that could possibly be the matching question $Q^*$ for the SMS query $S$.

Similarity score

- The similarity score is calculated for all the questions in $Q$-POSS.
- In an iterative manner, we select a word from the question $q$ which has maximum weight with respect to an SMS token $s$ and add its weight to the similarity score for $q$. That word is then removed from the list $Q$-words. This process is repeated till the word for each SMS token.

$$\text{SimilarityScore}(q)$$
$$\ldots(6)$$

Where, $w^* =$ word in the question $q$ with max
Weight w.r.t. SMS token $s_i$
Total Score

- The total score for a question q in Q-POSS is calculated by adding its keyword score and similarity score and is kept along with q.
- The question with the maximum total score is returned as the match Q*, for the SMS query S.

\[
\text{TotalScore}(q) = \text{KeywordScore}(q) + \text{SimilarityScore}(q)
\]

...(7)

Experiments and Results

- The system returns up to top 5 matching questions from the FAQ set with Q* as the best match for the SMS query S.
- If all matching questions for an SMS query had a total score lower than a threshold total score, then the query was considered to be irrelevant and "NULL" is given as output.
- Use of IDF gave more accurate results.
- Use of hash tables has made the system more time efficient.
Results contd..

- This system also proved to be more efficient than Python’s fuzzy match.
- The following results were obtained for this system:
  In Domain correct: 396/728 (0.54395604)
  Out of Domain correct: 1940/2677 (0.7246918)

Mean Reciprocal Rank (MRR): 0.8630503

Conclusion and Future Work

Thus, this system gives a smart and efficient algorithm for answering Frequently Asked Questions that are asked in SMS language. As future work I would like to address the following issues:
- Using a synonym dictionary that can add similar meaning words to the variant list for an SMS token.
- Improving the accuracy of the system with respect to in-domain queries.