Document classification with prerequisite analysis

Arun Balchandran, Rahul Singh and Suhas Patil

The University of Texas in Arlington

arunarunachalam.balchandran@mavs.uta.edu, suhas.patil2@mavs.uta.edu &

rahul.chouhan@mavs.uta.edu

Motivation

While going through research papers, students give up halfway because they couldn’t understand the concepts presented in the paper and hence waste valuable time. Most of the students pursuing a graduate degree or even an undergraduate degree for that matter come across the arduous task of reading and understanding research papers. Most of their productive time spent while going through a paper is the time it takes them to search and understand the concepts given in the paper that the students don’t understand. We propose to reduce this time wastage by making a one stop platform for learning research papers.

Related Work

We employ the use of document classification and prerequisite analysis to present users with the concepts presented in a paper.

Document classification can be done using a variety of different algorithms:

Expectation maximization:

The [1]EM algorithm is used to find (locally) maximum likelihood parameters of a statistical model in cases where the equations cannot be solved directly. Typically these models involve latent variables in addition to unknown parameters and known data observations. That is, either missing values exist among the data, or the model can be formulated more simply by assuming the existence of further unobserved data points.

Naïve Bayes Classification:

[2]Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. It is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features

TF-IDF:

In information retrieval, [3]tf–idf, short for **term frequency–inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is often used as a weighting factor in information retrieval, text mining, and user modeling. The tf-idf value increases proportionally to the number of times a word appears in the document, but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

Support Vector Machines:

In machine learning, [4]support vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier.

Online Document Classification:

Documents can also be classified online by using web apps that take as input text and convert that input into output that we desire such as the classification of the input.

E.g.: https://www.uclassify.com/browse/uclassify/topics

For prerequisite recognition there is some literature including :

Online Extraction APIs:

These are APIs that tell us the concepts present in a piece of text by analyzing the text in the backend.

E.g.: http://aylien.com/concept-extraction.

Research paper organizer:

Mendeley is an example of a research paper organizer that we plan of implementing later as an additional feature if time permits which will allow us to store the data that the user uploads and save and retrieve it for later use. This will allow the user to have a personalized library system.

Wikipedia API for querying data:

The Wikipedia API can be used to get data from Wikipedia for concept extraction and analysis for generation of the graph that we display to the user containing the nodes i.e. the concepts present in that research paper.

Systems Architecture

The system architecture of our system comprises of the following modules:

1. The web interface
2. Document Classifier
3. Prerequisite Analyzer
4. Paper Difficulty Classifier

The web interface:

The web interface is how a user interacts with the system and uploads a paper so that he/she can get a classification of the document with prerequisite analysis. The UI also provides the user with helpful links and a list of resources to study the given list of analyzed topics.

Document Classifier:

The document classifier tells us the general subject/s under which a document is classified.

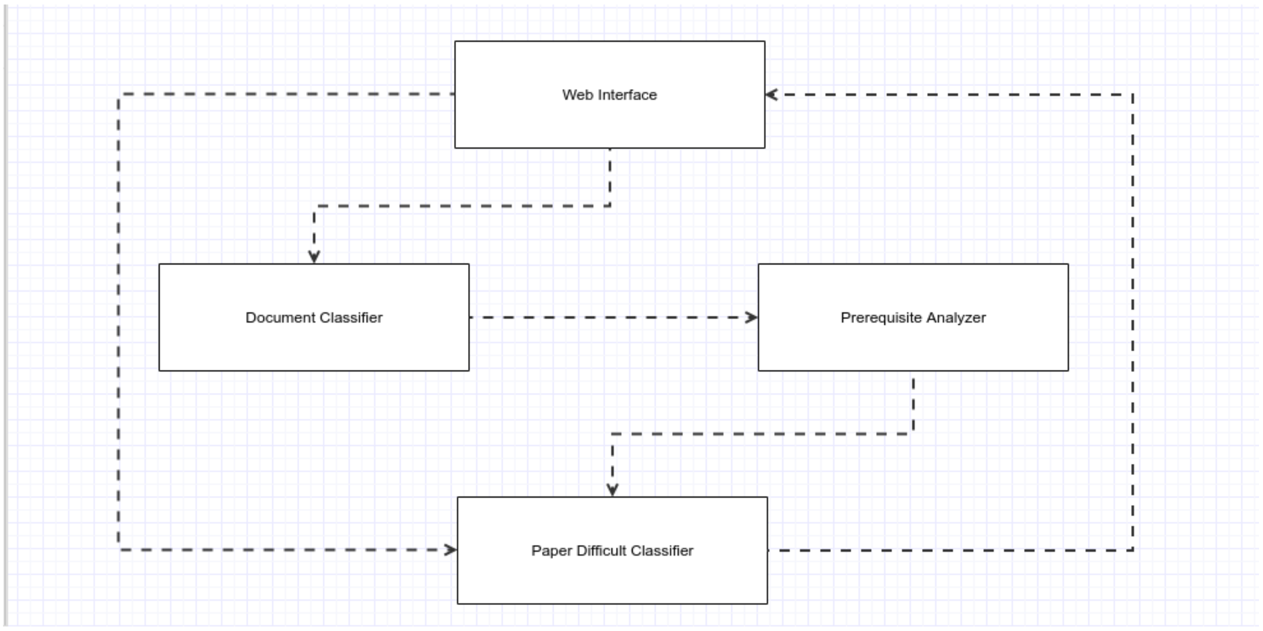
Prerequisite Analyzer:

The prerequisites analyzer give us a list of prerequisites detected from the content of the research paper. It will use the algorithm outlined in this paper

Paper Difficulty Classifier:

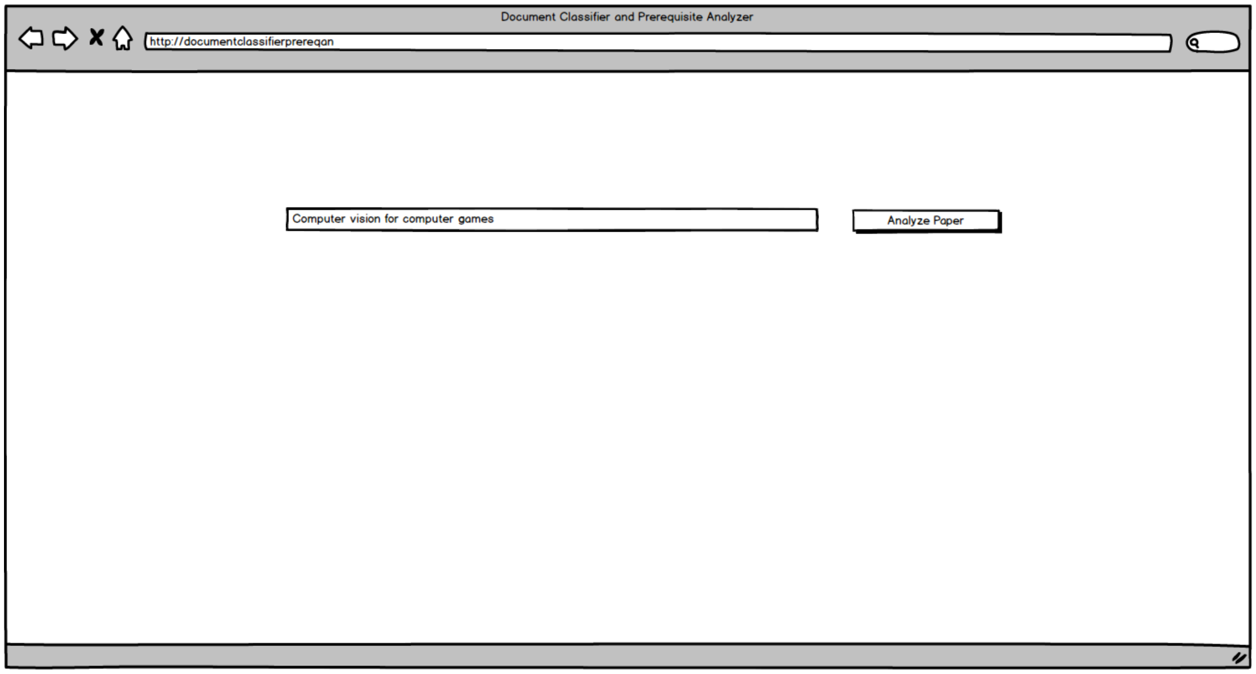
The paper difficulty classifier tells us the complexity of the research paper based on a user vote count metric.

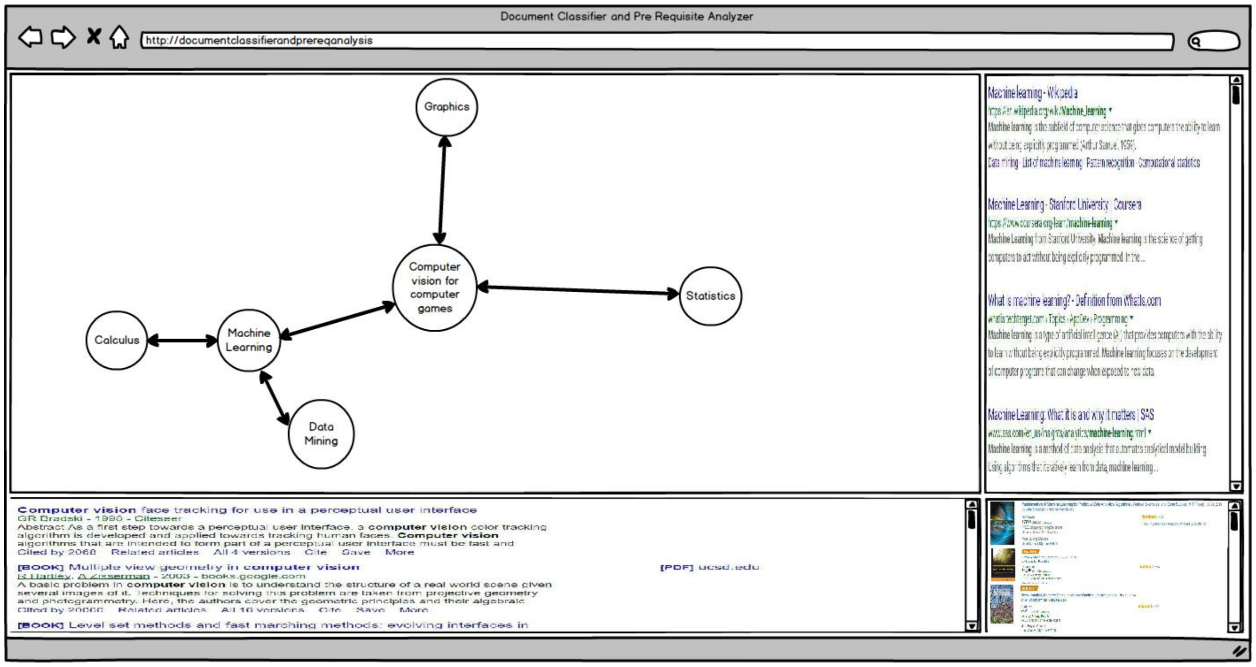
Here’s a diagram that will help you visualize the architecture:



Interface Design

The following is a diagram of the interface design using Balsamiq to make the mockups.


Document Classifier and Prerequisite Analyzer
http://documentclassifierprereqan
Computer vision for computer games
Analyze Paper


Document Classifier and Pre Requisite Analyzer
http://documentclassifierandprereqanalysis
Graphics
Computer vision for computer games
Statistics
Calculus
Machine Learning
Data Mining

Machine learning - Wikipedia
https://en.wikipedia.org/wiki/Machine_learning ▾
Machine learning is the subfield of computer science that gives computers the ability to learn without being explicitly programmed (Arthur Samuel, 1959).
Data mining · List of machine learning · Pattern recognition · Computational statistics

Machine Learning - Stanford University | Coursera
https://www.coursera.org/learn/machine-learning ▾
Machine Learning from Stanford University. Machine learning is the science of getting computers to act without being explicitly programmed. In the ...

What is machine learning? - Definition from WhatIs.com
whatis.techtarget.com › Topics › AppDev › Programming ▾
Machine learning is a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine learning focuses on the development of computer programs that can change when exposed to new data.

Machine Learning: What it is and why it matters | SAS
www.sas.com/en_us/insights/analytics/machine-learning.html ▾
Machine learning is a method of data analysis that automates analytical model building. Using algorithms that iteratively learn from data, machine learning ...

Computer vision face tracking for use in a perceptual user interface
GR Bradski - 1998 - Citeseer
Abstract As a first step towards a perceptual user interface, a **computer vision** color tracking algorithm is developed and applied towards tracking human faces. **Computer vision** algorithms that are intended to form part of a perceptual user interface must be fast and
Cited by 2066 Related articles All 4 versions Cite Save More

[BOOK] Multiple view geometry in **computer vision** [PDF] ucsd.edu
R Hartley, A Zisserman - 2003 - books.google.com
A basic problem in **computer vision** is to understand the structure of a real world scene given several images of it. Techniques for solving this problem are taken from projective geometry and photogrammetry. Here, the authors cover the geometric principles and their algebraic
Cited by 20666 Related articles All 16 versions Cite Save More

[BOOK] Level set methods and fast marching methods: evolving interfaces in

Requirements

The list of requirements for the project may be divided into 2 categories mentioned below.

Functional Requirements:

1. Informative interface to learn research paper which includes links to other papers, list of related papers for the given research paper.
2. Suggestion of difficulty level of the paper by getting inputs from the user.

Technical Requirements:

1. Web interface: HTML and Javascript can be used to build the front end.
2. The backed for the project will require a backend written in Python with Machine Learning, APIs, [6]Parser (Scholar.py) and the Wikipedia Dump for the prerequisite analyzer.

Typical Users:

1) Novice students who have limited exposure to research.
2) Users who need a quick refresher for a research paper.

Team Contributions and Timelines

Phase 1 - Survey/Requirement Gathering

1. Survey : 2-3 days
2. Selection of classification algorithm : 1 week
3. Selection of concept extraction algorithm and prerequisite analyzer : 1 week (in parallel)
4. UI library selection :
   1 week (in parallel)

Estimated completion time : 22th Feb.

Phase 2 - Development/Integration

1. Development and unit testing of selected classification algorithm : 2.5 weeks
2. Development and unit testing of selected prereq. analyzer : 2.5 weeks (in parallel)
3. UI design : 2 week (in parallel)
4. Integration of document classifier and prerequisite analyzer with UI : 1 week
5. Addition of user voting module (for paper difficulty) : 3-4 days

Estimated completion time : 31th March

Phase 3 - Post development task

1. Testing system after integrating modules and bug fixing : 2 week
2. Conducting studies to test efficiency of system : 1 to 2 weeks
3. Iterative improvement until deadline

Estimated completion time : 16th April.

Contribution :

- Suhas: UI, Integration of features
- Rahul: Classification using Machine Learning, User vote counting
- Arun: Prerequisite Analyzer, Extracting concepts from research papers

References

[1] https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

[2] https://en.wikipedia.org/wiki/Tf%E2%80%93idf

[3] https://en.wikipedia.org/wiki/Support_vector_machine

[4] https://www.uclassify.com/browse/uclassify/topics

[5] http://www.emnlp2015.org/proceeedings/EMNLP/pdf/EMNLP193.pdf

[6] http://github.com/ckreibich/scholar.py