Web Mining: Understanding the World Wide Web

An Introduction to Web Mining and Its Applications

Zime, Songbian PhD

September 19, 2023

The Web as the Largest Information Source

- The Web has become the largest source of information in the world.

Some Questions

- What can we learn from this data?
- What kind of services can we build around it?
- What insights and knowledge are hidden in the patterns?
- Does the data tell us something about our behavior?
- Can it answer sociological/psychological questions?

Definition of Web Mining

Web mining is the use of data mining techniques to automatically discover and extract information from Web documents/services (Etzioni, 1996).

Types of Web Mining

- **Web Content Mining:** Extract knowledge from web documents and user content.
- **Web Structure Mining:** Analyze the graph structure of the web and identify patterns.
- **Web Usage Mining:** Analyze the usage of web systems and extract patterns (e.g., navigation paths).

Google Flu Trends

- Can we predict flu outbreaks from search behavior?

Trends (http://www.google.org/flutrends/de/#DE)

Fields of Application

- Web Search
- Personalization in E-Commerce (e.g., Amazon)
- Recommender Systems
- Advertisement/Tracking User Behavior
- Digital Libraries
- Web Site Optimization
- Social Media Analytics (e.g., for Marketing)
- Sociology Research

A Small Use Case Example - Personal Filtering of Twitter Messages

- Twitter - a social network for short messages
- Let's consider a small use case: filtering Twitter Messages (Tweets) according to our interest.

The Program's Objectives

1. Tweets should be crawled from Twitter.
2. A user somehow must express what is relevant and what is not relevant for them.
3. Our program needs to represent this notion of relevance somehow.
4. Our program should store the relevant tweets in the user's inbox/PC.
5. Our program should allow the user to give feedback on what tweet is relevant/not relevant and adapt its strategy.
6. Our program should group relevant tweets together according to their topic.
7. Our program should show us the development of a topic over time.

Sketching the Solution - Step 1: Data Gathering

- Crawling/Gathering Data from Twitter.
- In the case of Twitter, it is easy since they provide an [API](https://dev.twitter.com/).
- Crawling arbitrary web pages requires writing more complicated crawlers and considering different formats.

Sketching the Solution - Step 2: Machine Learning - Classification

- By using Machine Learning methods, we can automatically classify tweets belonging to a certain class.
- So-called **Supervised Learning Methods** learn rules (e.g., mathematical models) from those examples to classify future, yet unknown examples.

Sketching the Solution - Step 3: Machine Learning - Clustering

- Clustering belongs to the **Unsupervised Machine Learning** methods.
- They group similar items together while maintaining maximal dissimilarity between groups.

Sketching the Solution - Step 4: Visualizing Data

- After obtaining topics in the form of groups of similar tweets, we need to display the number of tweets per topic over time.
- Visualization can help us understand the data better.

Digging Deeper - How to do the Heavy Lifting?

- Machine Learning is the core of our application. So how could this work?

A Linear Model for Classifying Tweets

- Let's assume relevance of a tweet only depends on the word occurring in a tweet.
- So a model for classifying tweets could be simply to:
  1. Assign every word a relevance score.
  2. Sum up all the relevance scores for every word in a tweet.
  3. If the sum exceeds a certain threshold, the whole tweet must be relevant.

More formally

- Let T be a list of tweets with denoting t_j as the j-th tweet.
- Let D be the dictionary of all words in all tweets.
- Let $d_i \in D$ be the i-th word in the dictionary.
- Let $w_i \in W$ be the relevance of d_i. Here, $w_i > 0$ denotes positive relevance, and $w_i < 0$ denotes negative relevance.
- Let $f_{i,j}$ denotes how often d_i occurs in t_j.

Representing Tweets as a Matrix

- For a more compact representation, we can represent each tweet as a vector of word occurrences and all tweets as a matrix.
- Classifying becomes vector/matrix operations.
- Obtaining this matrix representation is called "preprocessing."

How to Obtain the Weights?

- Trying to set the weights manually can be tedious and impractical.
- We need manually labeled data and an algorithm to "learn" the weights based on the labeled data.

Representing Tweets as a Matrix

- For a more compact representation, we can represent each tweet as a vector $\overline{f_j}$ of word occurrences and all tweets as a matrix F.
- Hence, classifying becomes vector/matrix operations:

$$F \cdot \overline{w} - \theta > 0$$

where $\overline{w}$ is the vector of relevance weights (note that strictly mathematically speaking, θ must be a vector). Most machine learning methods utilize such a matrix representation. Obtaining this matrix representation is called "preprocessing."

As a Consequence

- For successfully applying web mining techniques, we need to:
  1. Convert our data to matrices and vectors (**preprocess data**).
  2. Apply matrix operations and linear algebra (**numerical computations**).
- But how do we obtain the weights?
  - Trying to set the weights manually can be tedious at best and impractical in most cases.
  - We need to have:
    1. Manually labeled data, i.e., Tweets, that are considered to be relevant and some non-relevant ones.
    2. An algorithm to "learn" the weights based on the manually labeled data.

Adjusting the Weights - Error Calculation

Observation

- If relevance$(t_j) - \theta > 0$ but t_j is not relevant to the user, it means the relevance w_i for all words d_i in t_j was too high and needs to be reduced.
- If relevance$(t_j) - \theta < 0$ but t_j is relevant to the user, it means the relevance w_i for all words d_i in t_j was too low and needs to be increased.

Error Calculation

- Let y_j be defined as 1 for a relevant tweet and 0 otherwise.
- We can define the error as:

$$\text{error}_j = \text{classify}(t_j) - y_j$$

Weight Adjustment

- To adjust the weight for tweet t_j, we can use the following update rule:

$$w_i = w_i - \eta \cdot \text{error}_j \cdot f_{i,j}$$

where $\eta \in [0, 1]$ is a small constant called the learning rate.

y_j	classify(t_j)	New w_i
1	0	Increase
0	1	Decrease
0	0	No Update
1	1	No Update

Weight Adjustment - Updating Weights

- Doing this for all tweets $t_j \in T$ updates the weights correspondingly. You might need to do that during several iterations.
- There are numerous methods for solving this more efficiently and accurately. We will review some of those methods during this course.
- The algorithm sketched here is called stochastic gradient descent, and the model is a linear model. More details will be discussed later on.
- Every supervised learning algorithm needs feedback to adjust its model accordingly. In nearly all cases, these are single examples labeled with the target value.
- Manually labeled data is in most settings the scarcest resource.

Extracting Topics - Clustering Tweets

- We can distinguish between relevant and non-relevant tweets. But how to group the relevant tweets into single topics?
- First, we need to define what a topic is:
- We define a topic as a group of tweets similar in content.
- So how would you determine the similarity between two tweets?

Calculating the Similarity Between Tweets

- Observation: Two tweets that are similar share similar words.
- Hypothesis: The more words are shared, the higher the similarity between two tweets.

Formally

Let $\overline{f_j}$ be the word frequency vector for a tweet. The similarity between two tweets t_j and t_i can be calculated as:

$$\text{sim}_{j,i} = \overline{f_j}^T \cdot \overline{f_i}$$

In order to get a measure in the interval $[0, 1]$, we need to scale both vectors to unit length:

$$\text{sim}_{j,i} = \frac{\overline{f_j}^T \cdot \overline{f_i}}{\|\overline{f_j}\| \cdot \|\overline{f_i}\|}$$

This measure is also known as the **cosine measure** between two vectors. It measures the cosine of the angle between two measures and is widely used in web mining, especially for sparse vector spaces.

Clustering - Grouping Similar Tweets

We can now estimate the similarity of two tweets. But how do we group similar tweets, and can a tweet be in more than one group?
To simplify the task, we make the following assumption: Each tweet belongs to one group or topic.

Finding Groups of Similar Tweets

Given the similarity measure, how can we find groups of similar tweets?
Idea:

1. Choose k tweets representing different topics as a starting point.
2. Calculate the similarity between all tweets $t_j \in T$ and all k representative tweets.
3. Assign tweet t_j to the topic with the highest similarity.

This is the first step in the well-known **k-means Clustering Algorithm**.
There are many more clustering algorithms, and we will explore some of them along with their properties.
But, do you know why Steps 1-3 are not sufficient for finding a good grouping?

What I Offer to You

We may not be able to cover all aspects of web mining, but the goal is to assist you in taking your first steps in mining the web.
Mining involves extracting new, potentially interesting, and valid patterns from web sources.

Lecture Overview

There are numerous tools and technologies available for web mining. In this course, we will leverage the Python programming language and its wealth of libraries.
To equip you with the necessary skills for web mining with Python, I will deliver lectures and practical exercises on the following topics:

1. A short introduction to Data Science with Python
2. An introduction to the Python Programming Language
3. Important libraries for Data Science (along with the minimum necessary underlying theory)
    - Crawling Twitter and Cleaning HTML
    - Preprocessing Text: NLTK - Natural Language Toolkit
    - Numerical Computation: Numpy - Efficient Numerical Computations
    - Machine Learning: Scikit Learn - An easy-to-use machine learning library
    - Visualization: Matplotlib

Lecture Overview

We will focus mostly on practical aspects. Theory (e.g. algorithmic details) is only covered where necessary.However, I recommend that you also start obtaining the theoretical background,especially in machine learning. Only with a deep theoretical understanding you will be able to truely master the subject.

Your Projects

Experience is the best teacher. To truly master the subject, you need to undertake a web mining project on predefined topics.

Project Topics:

Our primary data source will be Twitter, simply because it provides an easily accessible data source. Additionally, web pages in the form of the WebKB dataset will also be available.

The following topics will be available:

1. Automatic Shitstorm Detection in Twitter (Group of 2-3)
2. Revealing topic, event, and person distributions in Twitter (Group of 2-3)
3. Sentiment analysis on Twitter Data (Group of 2)
4. Web page classification (Group of 1-2)

Each topic is associated with a set of questions that you should address in your project.

How - Timeline

Timeline:

- October to December: Building up the basics. Lectures by me supplemented with small exercises for you.
- December 5th: Group and topic assignment.
- December to January: Time to work on your project.
- End of January: Report your findings to your colleagues in a brief presentation (10 minutes).

All details are provided in Stud.IP.

Grading

Grading is based solely on your project:

- How accurate can you solve the task at hand?
- How good do your results generalize?
- What are the most important properties of the data to solve your task at hand?
- What is the most suitable algorithm and why?
- How correct is your justification of all the above points?

Since you can work in groups, you must make clear how you fulfilled which role. The following roles can be foreseen:

- Data Gathering and Preprocessing: Fetch the data and bring it in a usable form.
- Data Analysis Process: Set up the data analysis process by selecting algorithms and methods.
- Evaluation: Set up the evaluation procedure.

Grading

Grading is based solely on your project:
The single roles are intertwined and depend on each other.
All your results must be summarized in an IPython Notebook. I will have a talk with every group regarding the contents of the notebook,in order to determine the quality and correctness of your results and to make sure the work has been done by you as claimed.

Rules

- There is no mandatory attendance: If you already know a particular topic or if you want to learn it on your own, you don't have to participate in the weekly lecture. Note: You are learning for yourself, not for me.
- Teamwork yes, plagiarism no: Teamwork and discussing problems is important. However, copying code is forbidden and counts as plagiarism. If I encounter plagiarism in a group, all members will be graded with 5.
- You must be able to explain your work to me: If one member of a group cannot explain to me what he/she did in detail, I consider the work has been done by somebody else and will grade it as 5.
- Asking questions is not forbidden: Please, have the courage to ask questions if you are not understanding something.
- **The most important rule:** Have fun digging into data.

Recommended Readings

- Machine Learning, T. Mitchell, McGraw Hill 1997
  `http://www.cs.cmu.edu/~tom/mlbook.html`
- Pang-Ning Tan, Michael Steinbach, Vipin Kumar, Introduction to Data Mining, 2006, Pearson Education
- **Recommended Readings for a Short Introduction to Web Mining**
  Raymond Kosala and Hendrik Blockeel. 2000. Web mining research: a survey. SIGKDD Explor. Newsl. 2, 1 (June 2000), 1-15. DOI=10.1145/360402.360406
  `http://doi.acm.org/10.1145/360402.360406`
- Further, individual material will be presented within every lecture.
- You can also attend the Course "Machine Learning and Data Mining" or "Text Mining" for obtaining a deeper theoretical foundation.