CS 589 Fall 2020

Text Mining and Information Retrieval

Instructor: Susan Liu
TA: Huihui Liu

Stevens Institute of Technology
Welcome to CS589

• **Instructor:** Susan (Xueqing) Liu
• **Email:** xliu127@stevens.edu
• **CAs:**
  • Huihui Liu hliu79@stevens.edu
Who am I?

- Assistant professor joined Jan 2020
- PhD@UIUC 2019
- My research:
  - Helping users (especially software developers) to more quickly search for information

Diagram:
- Software engineering, security
- ML
- Text mining/IR
- My research
What is CS589 about?

• Text Mining
  • The study of extracting high quality information from raw texts

• Information retrieval
  • The study of retrieving relevant information/resources/knowledge to an information need
“Because the systems that are accessible today are so easy to use, it is tempting to think the technology behind them is similarly straightforward to build. This review has shown that the route to creating successful IR systems required much innovation and thought over a long period of time. “

— The history of Information Retrieval Research, Mark Sanderson and Bruce Croft
Information Retrieval Techniques

How does Google know cs 589 refers to a course?

How does Google know stevens = SIT?
Information Retrieval Techniques

Getting enough coverage of users’ information need

Making sure the results are returned to users fast

Query understanding, personalization, results diversification, result page optimization, etc.
A Brief History of IR

300 BC
Callimachus: the first library catalog

1950s
Punch cards, searching at 600 cards/min

1958
Cranfield evaluation methodology; word-based indexing

1960s
building IR systems on computers; relevance feedback

1970s
TF-IDF; probability ranking principle

1980s
TREC; learning to rank; latent semantic indexing

1990 - now
web search; supporting natural language queries;
**Information need**

**information need**

“An individual or group's desire to locate and obtain information to satisfy a need”, e.g., question answering, program repair, route planning

**query**

A (short) natural language representation of users’ information need
The Boolean retrieval system
The Boolean retrieval system

- e.g., `SELECT * FROM table_computer WHERE price < $500 AND brand = "Dell"
- Primary commercial retrieval system for 3 decades
- Many systems today still use the boolean retrieval system, i.e., faceted search
  - Library catalog, eCommerce search, etc.

- **Advantage**: Returns exactly what you want

- **Disadvantage**:
  - can only specify queries based on the pre-defined categories
  - two few / two many queries
The Boolean retrieval system

The user may specify a condition that does not exist
The Cranfield experiment (1958)

- Imagine you need to help users search for literatures in a digital library, how would you design such a system?

```
query = "subject = AI & subject = bioinformatics"
```

system 1: the Boolean retrieval system
Imagine you need to help users search for literatures in a digital library, how would you design such a system?

**The Cranfield experiment (1958)**

- Imagine you need to help users search for literatures in a digital library, how would you design such a system?

```
<table>
<thead>
<tr>
<th></th>
<th>intelligent</th>
<th>applications</th>
<th>creates</th>
<th>business</th>
<th>processes</th>
<th>bots</th>
<th>are</th>
<th>i</th>
<th>do</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc 1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Doc 2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Doc 3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
```

**query** = “artificial intelligence”

**bags of words representation**

**system 2: indexing documents by lists of words**
The Cranfield experiment (1958)

system 1

system 2

Boolean retrieval system < word indexing system
Word indexing: vector-space model

- Represent each document/query as a vector
- The similarity = cosine score between the vectors
Term frequency

\[
\begin{array}{c|cccccccccc}
\text{Doc 1} & \text{intelligent} & \text{applications} & \text{creates} & \text{business} & \text{processes} & \text{bots} & \text{are} & i & \text{do} & \text{artificial} \\
2 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
\text{Doc 2} & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\hline
\text{Doc 3} & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\[
tf(w, d) = \text{count}(w, d)
\]

\[
d_i = [\text{count}(w_1, d_i), \ldots, \text{count}(w_n, d_i)]
\]

- \(d_1 = [2, 1, 1, 1, 1, 0, 0, 0, 0, 0]\)
- \(d_2 = [1, 1, 0, 0, 0, 1, 1, 0, 0, 0]\)
- \(d_3 = [0, 0, 0, 1, 0, 0, 0, 1, 1, 1]\)

- query = “business intelligence”
- \(q = [0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1]\)
Vector space model

• To answer the query:
  - “business intelligence”
  - \( q = [0, 0, 0, 1, 0, 0, 0, 0, 0, 1] \)

\[
\text{score}(q, d) = \frac{q \cdot d}{\|q\| \cdot \|d\|}
\]

\( d1 = [2, 1, 1, 1, 1, 0, 0, 0, 0, 0] \)
\( d2 = [1, 1, 0, 0, 0, 1, 1, 0, 0, 0] \)
\( d3 = [0, 0, 0, 1, 0, 0, 1, 1, 1, 1] \)
TF-only representations is inaccurate

- Documents are dominated by words such as “the” “a”
- These words do not carry any meanings, nor do they discriminate between documents
  - q = “the artificial intelligence book”
  - d1 = “the cat, the doc, and the book”
  - d2 = “business intelligence”

\[
\begin{align*}
\text{score}(q, d_1) &= 0.8164 \\
\text{score}(q, d_2) &= 0.3535 \\
\Rightarrow \text{score}(q, d_1) &> \text{score}(q, d_2)
\end{align*}
\]
Zipf’s law distribution of words
Stop words

```r
> stopwords("english")

[ 1]  "i"            "me"            "my"            "myself"           "we"
[ 6]  "our"          "ours"          "ourselves"     "you"               "your"
[11]  "yours"        "yourself"      "yourself"      "he"                "him"
[16]  "his"          "himself"       "she"           "him"               "hers"
[21]  "herself"      "it"            "its"           "its"               "itself"
[26]  "them"         "their"         "theirs"        "them"              "they"
[31]  "which"        "who"           "whom"          "who"               "that"
[36]  "these"        "those"         "theirs"        "fails"             "are"
[41]  "was"          "were"          "been"          "being"             "being"
[46]  "have"         "has"           "had"           "having"             "do"
```
Desiderata for a good ranking function

- If a word appears everywhere, it should be penalized
- If a word appears in the same document multiple times, it’s importance should not grow linearly
- \( q = \text{“artificial intelligence”} \)
- \( d_1 = \text{““Artificial intelligence was founded as an academic discipline in 1955, and in the years since has experienced several waves of optimism”} \)
- \( d_2 = \text{““Artificial intelligence was founded as an academic discipline in 1955, artificial intelligence”} \)

\( d_2 \) is not twice more relevant than \( d_1 \)
Inverse-document frequency

- **Inverse-document frequency**: penalizing a word’s TF based on its document frequency

\[
IDF(w) = \log \frac{N}{df(w)}
\]

\[
q(d, w) = TF(d, w) \times IDF(w)
\]

- q = “the artificial intelligence book”
- d1 = “the cat, the doc, and the book”
- d2 = “business intelligence”

**TF-IDF weighting**

\[
\text{score}(q, d_1) = 0.8164 \rightarrow 0.2041
\]

\[
\text{score}(q, d_2) = 0.3535 \rightarrow 0.3535
\]

\[\Rightarrow \text{score}(q, d_1) < \text{score}(q, d_2)\]
Term frequency reweighing

• **Term frequency reweighing**: penalizing a word’s TF based on the TF itself

• If a word appears in the same document multiple times, it’s importance should not grow linearly

\[
tf(w, d) = \alpha + (1 - \alpha) \frac{\text{count}(w, d)}{\max_v \text{count}(v, d)}
\]

Max TF normalization

Log scale normalization

\[
tf(w, d) = \begin{cases} 
1 + \log \text{count}(w, d) & \text{count}(w, d) > 0 \\
0 & \text{o.w.}
\end{cases}
\]
Term-frequency reweighing

- **Logarithmic normalization**

**Log scale normalization**

\[
tf(w, d) = \begin{cases} 
1 + \log \text{count}(w, d) & \text{count}(w, d) > 0 \\
0 & \text{o.w.}
\end{cases}
\]

- \(q = \text{“the artificial intelligence book”}\)
- \(d_1 = \text{“the cat, the doc, and the book”}\)
- \(d_2 = \text{“business intelligence”}\)

\[
\text{score}(q, d_1) = 0.8164 \rightarrow 0.7618
\]

\[
\text{score}(q, d_1) = 0.3535 \rightarrow 0.3535
\]
Document length pivoting

• Another problem with TF-IDF weighting
  • Longer documents cover more topics, so the query may match a small subset of the vocabulary
  • Longer documents need to be considered differently

\[ q = \text{“artificial intelligence”} \]

\[ d_1 = \text{“artificial intelligence book”} \]

\[ d_2 = \text{“Artificial intelligence was founded as an academic discipline in 1955, and in the years since has experienced several waves of optimism”} \]

\[ \text{score}(q, d_1) > \text{score}(q, d_2) \]
Document length pivoting

- For each query q and each document d, compute their relevance score \( score(q, d) \)

- Manually evaluate the relevance between q and d

\[
\text{relevance judgment}@l = \frac{\text{count}(\text{length} = l, \text{rel} = 1)}{\text{count}(\text{length} = l)}
\]
Document length pivoting

- Rotate the relevance score curve, such that it most closely align with the relevance judgement curve

\[ y = x \]

\[ pivot = pivot \times slope + intercept \]

\[ pivoted\_normalization = (1.0 - slope) \times pivot + slope \times oldnormalization \]
Document length pivoting

- Rotate the relevance score curve, such that it most closely aligns with the relevance judgement curve

\[
\frac{\text{tf} \cdot \text{idf weight}}{(1.0 - \text{slope}) \times \text{pivot} + \text{slope} \times \text{old normalization}}
\]

\[
1 + \frac{\text{slope}}{(1.0 - \text{slope}) \times \text{pivot}} \times \text{old normalization}
\]

the similar formulation will be frequently used later
More on retrieval model design heuristics

- Axiomatic thinking in information retrieval [Fang et al., SIGIR 2004]

<table>
<thead>
<tr>
<th>Constraints</th>
<th>Intuitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>TFC1</td>
<td>to favor a document with more occurrence of a query term</td>
</tr>
<tr>
<td>TFC2</td>
<td>to favor document matching more distinct query terms</td>
</tr>
<tr>
<td>TFC3</td>
<td>to make sure that the change in the score caused by increasing TF from 1 to 2 is larger than that caused by increasing TF from 100 to 101.</td>
</tr>
<tr>
<td>TDC</td>
<td>to regulate the impact of TF and IDF</td>
</tr>
<tr>
<td>LNC1</td>
<td>to penalize a long document (assuming equal TF)</td>
</tr>
<tr>
<td>LNC2, TF-LNC</td>
<td>to avoid over-penalizing a long document</td>
</tr>
<tr>
<td>TF-LNC</td>
<td>to regulate the interaction of TF and document length</td>
</tr>
</tbody>
</table>
IR != web search

• The other side of information retrieval techniques
  • Recommender systems (users who bought this also bought…)
  • Online advertising
IR $\neq$ web search

- Reasoning-based question answering systems
What about text mining?

Text Mining
- Data mining
  - document classification
  - document clustering
- AI/ML
  - information extraction
- NLP
- IR
  - sentiment analysis
  - web search & mining
- Database
Syllabus

- Vector space model, TF-IDF
- Probability ranking principle, BM25
- IR evaluation, query completion
- Inverted index, ES, PageRank, HITS
- Relevance feedback, PRF
- Neural IR

- EM algorithm
- RNN/LSTM
- Transformer/Bert

- Frontier topic: recommender system
- Frontier topic: opinion analysis/mining
- Frontier topic: NMT, program synthesis
Assignment goals

Upon successful completion of this course, students should be able to:

• Evaluate ranking algorithms by using information retrieval evaluation techniques, and implement text retrieval models such as TF-IDF and BM25;

• Use Elastic search to implement a prototypical search engine on Twitter data;

• Derive inference algorithms for the maximum likelihood estimation (MLE), implement the expectation maximization (EM) algorithm;

• Use state-of-the-art tools such as LSTM/Bert for text classification tasks
Prerequisite

• CS116 is required for undergrad, CS225 is recommended (data structure in Java)

• Fluency in Python is required

• A good knowledge on statistics and probability

• Knowledge of one or more of the following areas is a plus, but not required: Information Retrieval, Machine Learning, Data Mining, Natural Language Processing

• Contact the instructor if you aren’t sure
Format

• Meeting: every Monday 8:15-9:45

• 4 programming assignments
  • Submit code + report

• 1 midterm
  • in class

• Final project
Final Project

**Oct 19 - Oct 26**  
Students choose a topic; for each topic, they pick 2-3 coherent papers, and write a summary for the paper.

**Oct 26 - Nov 16**  
Students who share the same interest are categorized into groups; each group propose a novel research topic motivated by their survey.

**Dec 14**  
Deliver a presentation in Week 14

**Dec 20**  
Submit their implementation (code in Python) as well as an 8-page academic paper as their final project.
Grading

- Homework - 40%, Midterm - 30%, Project - 30%

- Late policy
  - Submit within 24 hours of deadline - 90%, within 48 hours - 70%, 0 if code not compile
  - Late by over 48 hours are generally not permitted
    - Medical conditions
    - A sudden increase in family duty
    - Too much workload from other courses
    - The assignment is too difficult
Plagiarism policy

- We have a very powerful plagiarism detection pipeline, do not take the risk

- Cheating case in CS284
  - A student put all his homework on a GitHub public repo
  - In the end, we found 8+ students copied his code

Thu 5/21/2020 1:38 PM
To: Xueqing Liu

Hello Prof. Liu —

I see that you got an F in CS284C for S20. She has done well in all her other classes and found this F to be shocking. I have to reach out to her. It would help me if you can provide some feedback into what went wrong for her. Any feedback you can provide will be helpful.

Thank you,
Question answering

- Please do not ask your questions in Canvas, most questions can be asked on Piazza, otherwise use emails.
Question asking protocol

- Regrading requests: email TA, cc myself, titled [CS589 regrading]
- Deadline extension requests: email myself, titled [CS589 deadline]
- Dropping: email myself, titled [CS589 drop]
- All technical questions: Piazza
  - Homework description clarification
  - Clarification on course materials
- Having trouble with homework: join my office hour directly, no need to email me
  - If you have a time conflict, email me & schedule another time
- Project discussion: join my office hour
- Ask any common questions shared by the class on Piazza
Your workload

- **Aug**: First Day of Instruction
- **Sept**: Lectures/Readings
- **Oct**: Programming Assignments
- **Nov**: Midterm
- **Dec**: Project

- **Nov**: Thanksgiving
- **Dec**: Last Day of Instruction
Books

• No text books

• Recommended readings:
  • Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze, Introduction to Information Retrieval, Cambridge University Press. 2008