CS2309 Project Presentation: Web Crawling

Lim Jia Yee

November 11, 2016

Problem: All thanks to you.

- It is easy to create content on the web.
- It is easier for the web to expand now thanks to you.
- It is easiest to take web search for granted.

Motivation

- Ensure web search remains optimised; prevent world destruction.
- Vested interest in learning from data (i.e. machine learning).
- Brainchild: Exploring the possibility of enhancing web crawling with decisions based on data.

Relevance

- Educate on the considerations of designing a web crawler, or equivalent systems.
- Increasing the scope of machine learning as a solution.
- Target Audience: People who maintain focused web crawlers

Three-Part Solution

1. Requesting
2. Deciding
3. Parsing

Single Web Crawler Algorithm

1. Decide on a good hyperlink to begin crawling from.
2. Fetch the corresponding web page of the hyperlink in (1).
3. Parse for all hyperlinks and store them.
4. Process the contents of the web page.
5. From the storage of hyperlinks, extract an unvisited one.
6. Repeat from (2).

Graph Problem

- The World Wide Web is the graph.
- Directed, unweighted, unknown.
- Vertices: Web pages and their contents
- Edges: Hyperlinks

Graph Exploration

- Breadth-first search
- Depth-first search
- Iterative deepening depth-first search
- Beam search (i.e. enhanced best-first search)

Complexity Analysis

- Breadth-first search: Memory
- Depth-first search: Narrow scope
- Iterative deepening depth-first search: Data structure
- Beam search: Accuracy of the heuristic

Solution Part I: Graph Exploration

- Modified depth-first search
- Does not always push to the stack

Alert: Another Performance Bottleneck

- Fetching the web page and parsing it.
- Why not decide beforehand whether we should even do it?

Solution Part II: Reinforcement Learning

- Why?
- No training data.
- Unknown until experienced.

Markov Decision Process (MDP)

1. State
2. Action: To parse or not to parse, that is the question.
3. Reward
4. Policy

Reinforcement Learning: State

1. How relevant the current host is.
2. How relevant the previous host was.
3. How many web pages belonging to the current host were actually parsed, and not skipped.
4. How relevant the URL is.

Reinforcement Learning: State

- Real-valued states $\Rightarrow$ infinite states.
- Reduce dimension via intervals.
- Result: $2^4 = 16$ (high and low features) states.

Reinforcement Learning: Action

- Parse or not.

Reinforcement Learning: Reward

- Number of "high" features in the state.
- No additional penalisation of "low" features in the state.

Reinforcement Learning: Policy

- Value Iteration Algorithm
- Policy Iteration Algorithm

Value Iteration Algorithm

1. Assign random **true values** to each of the states.
2. For every state, calculate a new true value based on its neighbours' current true values.
3. Terminate if any of the true values in (2) changes by more than a user-specified δ. Else, repeat from (2).

Value Iteration Algorithm: Limitations

- Slow convergence.
- Do we want true value or policy?

Policy Iteration Algorithm

1. Create a random **policy** (i.e. assign a random action for each state).
2. Calculate the true value of each state given the policy in (1).
3. Based on these new true values, choose the optimal action for each state.
4. Terminate if none of the actions in (3) is changed. Else, repeat from (2).

Solution Part III: Parsing

- Not the main focus, but needed for testing.
- Define "relevant": Keywords in the web page match the list of "search words" prepared beforehand.

Rapid Automatic Keyword Extraction (RAKE)

1. Remove punctuation and special characters.
2. Remove stop words.
3. Stem the remaining words or phrases.
4. Find the degree of each word or phrase in (3).
5. Count the frequency of each word or phrase in (3).
6. Compute $score = \frac{degree}{frequency}$ for each word or phrase in (3).

RAKE: Specifications

- Assumes input is in standard English.
- Numbers are also extracted.

Similarity Measure: word2vec

- Words as vectors.
- Similarity $\Rightarrow$ distance between words.

word2vec: Limitations

- Bias towards exact words.
- Resolve by classifying a range as “similar”.

Three-Part Experiment

- Modified DFS
- Policy Iteration
- RAKE and word2vec

Modified DFS versus BFS

- Reduced execution time (approx. 40%)
- Reduced memory usage (approx. 40%)
- Not verified: “Quality” of the visited web pages
- Can be verified via logic, but the algorithm will have to be even more accurate in targeting particular kinds of URL.

Final Policy

- Policy was "False" for every state $\Rightarrow$ nothing will ever be parsed.
- Policy remains the same despite increase of γ, number of iterations, and increase in penalty for skipping.
- Conclusion:
  - Insufficient domain knowledge,
  - Should not define parameters by hand, and/or
  - Unsuitable library
- Try model-free learning instead.

Phrase Similarity: Generality

- Keywords which are more general than search words can still score very high in similarity.
- Discovered: “Relevance” and “similarity” are not the same.

Future Work: Beam Search and URL Analysis

1. Rank vertices by their edges (URLs).
2. Add only the first N ranked edges to the heap.

Conclusion

- Failure: Nope. Instead, you have *succeeded* in proving that this failed. Try something else.
- Failure: Is when you give up.