Agents explore a hyperlink graph of the web.
aim - distinct web pages.

Cora - domain specific search engine for computer science papers.
-resolves forward/backward references
-finds titles, authors, etc

Reinforcement Learning

state set -	$s \in S$
action -	$a \in A$
transition - T :	$S \times A \rightarrow S$
reward - R:	$S \times A \rightarrow \Re$
Discount factor -	$0 \leq \gamma \leq 1$ sooner reward is better
Value of each state -	$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t$Value of each state - $V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t$
Optimal policy - π^*	value of this is V^*
Optimal policy =	$\pi^*(s) = argmax Q^*(s,a)$

Value of selecting an action a from a state s

$$Q^*(s,a) = R(s,a) \text{ (reward) } + p\gamma V^*(T(s,a)) \text{ (best policy)}$$

.

Cheese Searching vs. Spidering

These are similar because there is an immediate reward.

Also because they consider the future reward.

State locs to be considered

Number of acts are large and dynamic.

Actually follow the link.

.

Why the Spidering Method Works

Able to speed up searches by looking for new keyword and adding to database later.

Goal Practicality

-Problems-:

-state space is $2\hat{(}$on topic)

-action space - number of URLs

---

state is independent of documents consumed

action distance by words in text use neighborhood text (bag-of-words)

Q function becomes a mapping from bag-of-words into a scalar (sum of future reward)

Two Subproblems

-Assigning Q values to hyperlinks (learn mapping from text to Q vals)

Simplest mapping -	1 to links pointing to paper
.	0 to others $(\gamma = 0)$

Slightly more complicated	.
Calculate the sum of the reward from the page	$\gamma \geq 0$

.

The order of paper retrieval has bearing on the results.
If the reward for A is 1 and the reward for B is 0 but it has links to 1000 papers, the best order for reaching these pages is first A then B for a reward of 101111..... instead of B then A for a reward of 011111..... This is because the earliest reward is always considered better.

Naive Bayes

document class	C_j
document frequency	$P(C_j)$
word	w_t
Vocab:	v
frequency classifier expects word to appear in docs	$P(w_t\|c_j)$
document	d_i

Assumptions to classifications
-words occur independently
-calculate probability of each class with evidence of documents

$$P(c_j|d_i)$$

kth word in document is

$$w_{d_{ik}}$$

good to learn the parameters - $P(C_j)$and$P(w_t|c_j)$

Method using set of labeled training documents.

Constant Model
-sum Q values class together
-place hyperlinks into bins corresponding to their Q values.
Run Bayes.

Determine Q values of unknown
-compute probability of classifying member of each bin
-compute weighted average of bin's average Q value

Experiments
Pitted four spiders against each other:

1. Breadth-first spider .
2. Immediate spider $\gamma = 0$
3. Future spider $\gamma > 0$
. still get reward, may lead to something in the future
4. Distance spider combination of 2 and 3

Consider time to get $\frac{1}{2}$ research papers
-simple, but may be harder to obtain second $\frac{1}{2}$

Facts
-distance spider was better earlier

-immediate spider was better in the longrun

Future Work

-number of links cooresponding to page of links, higher probability ratings of papers.

-if hyperlinks are found in two pages A and B, there should be a better way to keep track of which links were already traversed so that they are not traversed twice.

One thing found lacking in the paper was that the actual speed of these spiders was never discussed.

However, the point was of the research was to maximize the total number of papers regardless of speed.