Project Report for CS224W: Evaluating, Enabling, and Exploiting Wikipedia's Navigability

Robert West

December 7, 2010

1 Introduction

The motivation for this project is to revisit NLP-related research I have done previously [WPP09] by changing perspective and looking at it from a network-centric viewpoint. The contribution of my original research was devising the human-computation game 'Wikispeedia' (`wikispeedia.net`) and demonstrating that the data it generates can be exploited in order to define a measure of semantic distance between concepts. The game is played by a single human, who is given two random Wikipedia articles, and whose mission is to start on the first article and use Wikipedia's internal hyperlinks to click her way to the second one (the 'goal article'), minimizing the number of clicks. Players naturally have no knowledge of Wikipedia's hyperlink structure, yet they achieve short search times through locally greedy behavior, leveraging semantic background knowledge to gauge which outgoing links of the current article are most promising to decrease the distance to the goal. In other words, humans perform a decentralized search based on their internal *semantic distance measure*.

The game uses a Wikipedia version of reduced size, containing 4,604 articles [Wik07]. Since August 2008, I have collected around 27,000 game instances. In the original project, I introduced a method that effectively extracts the semantic commonsense knowledge players used in these games, thereby defining a semantic distance measure between concepts[1] (assuming a one-to-one mapping between Wikipedia articles and concepts). I now want to reinvestigate the data from the angle of information network analysis.

I will first concisely state the questions this work attempts to answer, which sets the stage for the remainder of the present report.

In class we talked about Kleinberg's model [Kle00], which implies that networks are efficiently navigable if link lengths (with respect to an underlying metric; e.g., geographical or, in our case, semantic distance) follow a power law distribution with a certain exponent. This motivates

Question 1 Are existing measures of semantic distance well suited to make Wikipedia (more precisely: its hyperlink graph) efficiently navigable?[2]

If so, it is trivial to create an agent that can successfully play Wikispeedia with no knowledge about its overall link structure, by simply performing greedy decentralized search with respect to a given measure of semantic distance. Pushing further, we may then ask

Question 2 Can we do better than that? That is, how much better than such a very simple greedy agent is an agent that learns an 'optimal' navigation policy from experience?[3]

In the rest of this report, I will show that the answer to both questions is Yes, as follows: Section 2 summarizes my findings; Section 3 provides an analysis of human performance, while Section 4 does so for an array of automated Wikispeedia-playing agents. Finally, Section 5 wraps up by sketching some future perspectives.

[1] This constitutes the 'exploiting' part of the title.
[2] This constitutes the 'evaluating' part of the title.
[3] This constitutes the 'enabling' part of the title.

2 Summary of Findings

I will provide fourfold evidence for the claim that Wikipedia is efficiently navigable:

1. Humans manage to achieve very short search times in Wikispeedia, their median search time being only two steps longer than the shortest path.

2. An automated Wikispeedia-playing agent performing decentralized search using off-the-self computational measures of semantic distance achieves vastly shorter search times than an agent performing plain BFS or DFS, and often even beats the median human.

3. Link lengths (with respect to the semantic distance measures) in Wikipedia are distributed in a way that makes the link graph efficiently navigable by decentralized search.

4. By learning a more sophisticated search policy from automated game-play, a computer player generally achieves short search times even when the straightforward decentralized-search agent fails, and significantly surpasses the performance of human Wikispeedia players.

Before providing details for each of these arguments, let me define some terms. A Wikispeedia *mission* is defined as a pair of a start and a goal node. There might be numerous game instances for the same mission. *Search time* is defined as the number of Wikipedia articles a player (human or automated) visits during a game. Note that it is legitimate to back up (using the browser's 'Back' button in the case of human players); search time is the number of all articles visited, including those from which the player has backed up.

3 Human Performance on Wikispeedia

To date, I have collected about 27,000 game instances of Wikispeedia. For the experiments presented here, I used the 1,000 most frequently played missions, which account for a combined 5,166 games. The minimum number of game instances per mission is 4, the maximum 25.

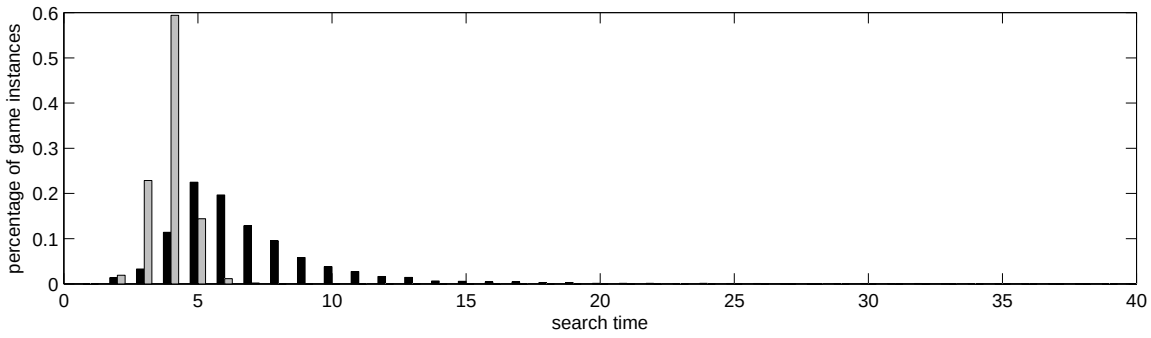

Figure 1: Histogram of search times for 5,166 games of Wikispeedia (1,000 distinct missions). *Black:* human players; five games (of length 41, 42, 51, 65, and 72, respectively) are not shown. *Grey:* lengths of shortest solutions for the same games.

Figure 1 contains histograms of human and optimal search time, where optimal search time is defined as the number of nodes along a shortest path from the start to the goal article. There are two important observations.

First, in most cases, human players achieve search times close to the optimum. The most likely human search time (5) is only one step longer than the shortest path length (4); for median human search time, the difference is 2 (6 vs. 4). This is remarkable since shortest-path finding requires knowledge of the entire link graph, which humans do not possess. Later, in Section 4.1, we will see that searching for the goal article using standard DFS or BFS typically takes two to three orders of magnitude longer.

Second, whereas shortest path lengths are narrowly bounded between 2 and 6 for the 1,000 test missions (the maximum across all potential missions is 9), human search time is distributed with a heavy

tail. This means that, although humans very often successfully find close-to-optimal paths, they also frequently take significantly longer. In fact, the tail follows a power law, as evident from Figure 2. I used the method described by Clauset *et al.* [CSN09] to estimate the parameters $\alpha = 3.50$ and $x_{\min} = 6$.

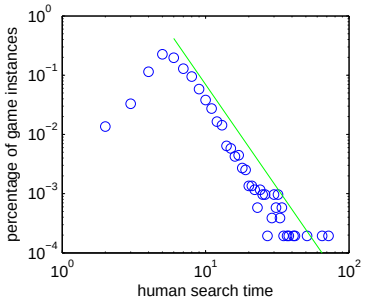

Figure 2: Log–log version of the histogram of human search times (cf. black bars in Figure 1). The green line shows the estimated power law with parameters $\alpha = 3.50$ and $x_{\min} = 6$.

4 Performance of Automated Wikispeedia Agents

4.1 Decentralized Search Using Off-the-Shelf Semantic Distance Measures

The previous section made the point that humans' internal semantic distance measure makes Wikipedia efficiently navigable. Next I want to investigate whether this is also the case for computational semantic distance measures.

In this context, a semantic distance measure is simply a function mapping pairs of concepts to $[0, 1]$. When we speak of semantic relatedness, we mean the complement, i.e., one minus distance. Several such relatedness measures are readily available, and we experimented with a few, achieving very similar results. For the sake of clarity we therefore restrict the discussion here to one particular relatedness measure. It represents a concept c_1 as the vector of Wikipedia links which the Wikipedia article about c_1 contains and quantifies the relatedness between c_1 and another concept c_2 as the cosine between their two respective vectors (cf. [MW08] for more details).

We have argued that, since humans do not have global knowledge about Wikipedia's link structure, they hop towards the goal article in a greedy decentralized-search fashion, always continuing to the current article's neighbor that minimizes the semantic distance to the goal. We can then program an agent that attempts to mimic human behavior in a very simple way: beginning with the start article, perform a depth-first search (DFS), iterating over the current article's neighbors in order of increasing semantic distance to the goal article. If humans indeed play Wikispeedia in a decentralized-search way, and if the computational semantic distance measure is any good, then the observed search times of the automated agent should (at least qualitatively) be similar to those achieved by humans.

Figure 3 suggests that this is in fact the case. The blue plot on the left-hand side is a log–log histogram of the number of steps the automated agent needs to solve the 1,000 test missions. Notice that the graph looks similar to the histogram of human search times (Figure 2), with the difference that the decentralized search sometimes takes very long, while the longest observed human search time is 72. When we discard all of the agent's searches that took longer than this longest human search (only 68 out of 1,000 searches), we obtain the histogram on the right of Figure 3, which resembles the distribution of human search times (Figure 2) a lot. Fitting a power law, we obtain the parameters $\alpha = 2.79$ and $x_{\min} = 4$. The larger exponent of humans ($\alpha = 3.50$) means that, as expected, longer searches are less likely for humans than for the automated agent.

To demonstrate that employing a semantic distance measure speeds up the search tremendously, let us compare the performance of the decentralized-search (DS) agent to that of two less sophisticated colleagues of his. A DFS agent is very similar to the DS agent, except that it has an F. Nah, I'm kidding, the real difference is that the DFS agent has no semantic knowledge and iterates over the current article's neighbors in random order rather than in order of increasing semantic distance to the goal. And a BFS agent is, well, an agent performing BFS. An important difference between DFS and BFS is normally

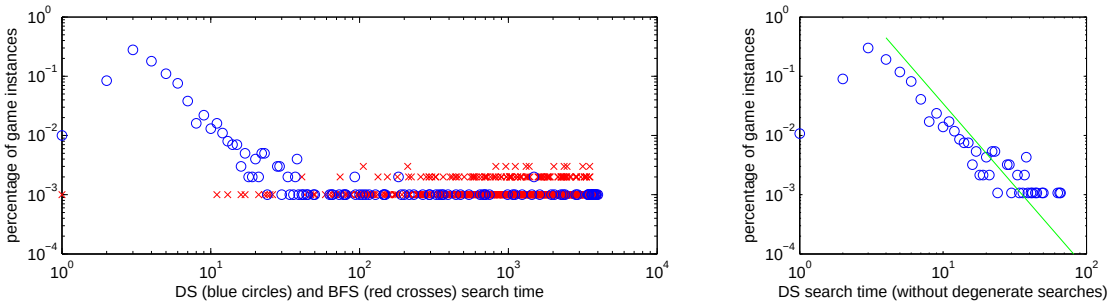

Figure 3: *Left/blue:* Histogram of search times achieved by the agent performing decentralized search to solve the 1,000 Wikispeedia test missions. *Left/red:* Histogram of search times achieved by a BFS agent. *Right:* Same as left/blue, but discarding all searches taking longer than 72 steps. The green line shows the estimated power law with parameters $\alpha = 2.79$ and $x_{\min} = 4$.

that BFS finds shortest paths, while we have no such guarantee for DFS. However, this difference plays no role for us, as we measure performance not in terms of length of the effective path found but in terms of nodes visited during the search.

The red histogram on the left-hand side of Figure 3 visualizes the search time of BFS in the 1,000 test missions. Clearly, there is no power law as in the case of DS. On the contrary, long search times seem to occur more frequently than short ones. That is, while the DS agent's search time distribution is qualitatively similar to that of humans, the more basic BFS agent's is not.

Next consider the scatter plots of Figure 4, which confirm that DS is superior to both DFS and BFS. In the following, I refer to a plot by its row and column numbers (in this order). Each plot compares the search times of two search algorithms A_1 and A_2; a point at position (x, y) represents game instances for which algorithm A_1 needed x and for which A_2 needed y time steps. A point being in the left upper half means that algorithm A_1 needed fewer time steps to solve the respective missions than algorithm A_2 (and *vice versa* if the point lies in the right lower half). With this in mind, consider plots $(2, 2)$ and $(3, 2)$. Clearly, DS outperforms both BFS and DFS on all but a few missions. As for the one-eyed who is king in the land of the blind, that would be BFS: although this is not immediately visible[4] in the relevant plot $(1, 2)$, on 635 of the 1,000 test missions, BFS found the goal in fewer time steps than DFS.

Let us now compare the automated agents to human players. Remember that each of the 1,000 test missions was solved in between 4 and 25 game instances. When we talk of human search time for a mission, we in fact refer to the median length of all games solving that mission. As expected, both DFS and BFS fare much worse than humans (cf. plots $(2, 1)$ and $(3, 1)$). It is more surprising that the DS agent in a sense outperforms the median human player. For 690 of the 1,000 test missions, the DS agent needs at most as many steps as the median human (in plot $(1, 1)$ this is not obvious because a point (x, y) represents all missions for which algorithm A_1 needed x and for which A_2 needed y steps, and there might be several such missions). In 567 missions, the DS agent is even strictly faster than the median human.

4.2 Analysis of Wikipedia's Link Length Distribution

What makes a simple greedy algorithm like decentralized search so successful at playing Wikispeedia? In class we talked about Kleinberg's model [Kle00], which implies that networks are efficiently navigable if link lengths (with respect to an underlying metric; e.g., geographical or, in our case, semantic distance) follow a power law distribution with a certain exponent. In this case, the expected hitting time of decentralized search is $O(\log^2 n)$, while it is $O(n^\beta)$ otherwise, where n is the number of nodes in the network and $\beta > 0$. Kleinberg's model assumes that all nodes are uniformly spread on a grid, with additional links drawn from the power law distribution. However, this is not the case with Wikipedia's link graph and semantic distance as the underlying metric.

Liben-Nowell *et al.* [LNNK+05] generalized Kleinberg's model to networks without an underlying grid structure by introducing 'rank-based friendship', where the power law is not over distance *per se*

[4]But hey, we're talking about one-eyed vs. blind.

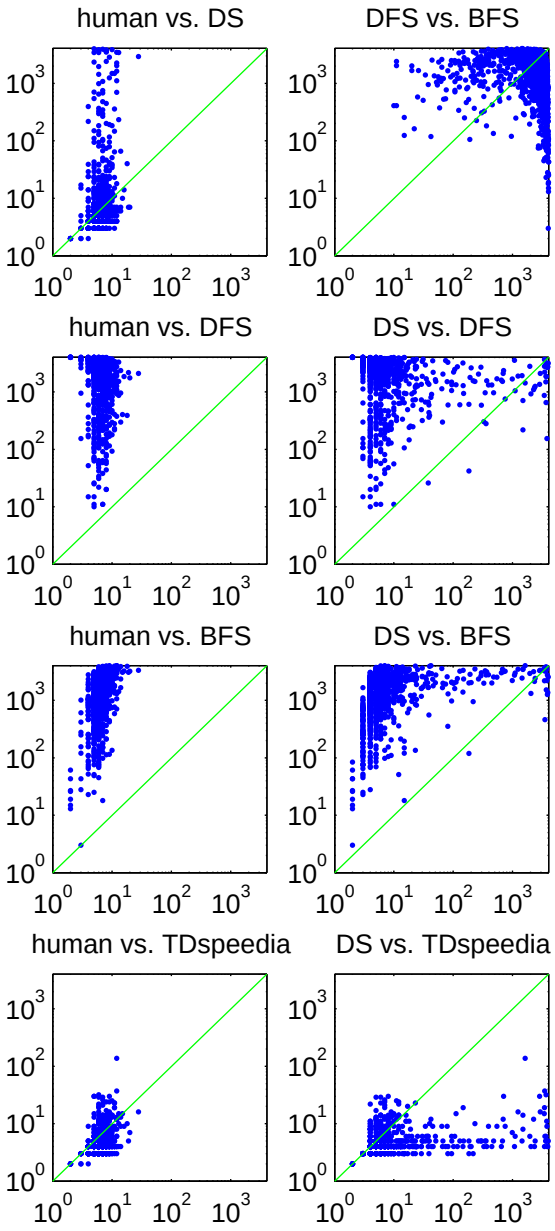

Figure 4: Scatter plots comparing different search algorithms for playing Wikispeedia.

but over the ranks induced by the distance. Given a node u, list all other nodes of the network in order of increasing distance from u. If v is the r-th node in that list, then v is said to have rank r (note that this is always conditional on u). Liben-Nowell *et al.* find that if an edge (u, v) is present in the network with probability proportional to r^{-1}, then the network is efficiently navigable.

Figure 5 contains a log–log histogram of the ranks corresponding to the links present in our Wikipedia version. If rank-based link length followed Liben-Nowell *et al.*'s model we would expect to see a straight line of slope -1. While this is not entirely the case, since the plot is slightly curved, we come pretty close. A straight line of slope -1 would, e.g., stretch from the upper left to the lower right corner of the plot, so the slope of our curve is close to -1.

I therefore conjecture that the good performance of the DS agent results from the fact that Wikipedia links are distributed in a fashion approximately inversely proportional to their ranks (ranks being derived

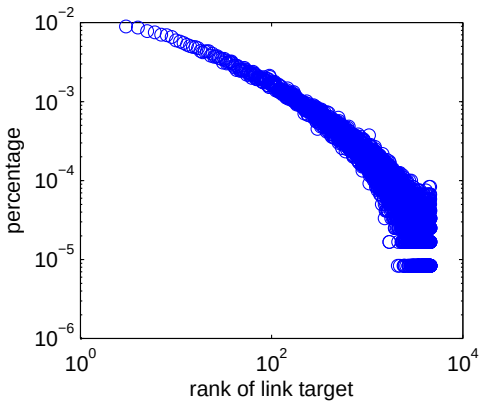

Figure 5: Rank-based link length distribution in our small Wikipedia version. Ranks are computed using the same semantic distance measure also used by the decentralized search agent.

from the semantic distance measure also leveraged by the DS agent), as stipulated by Liben-Nowell *et al.*'s model. This latter fact should not come as too much of a surprise since the observation that Wikipedia links bear semantic value lies at the very foundation of the game of Wikispeedia. Otherwise humans could not play the game solely based on their own semantic background knowledge and without knowing Wikipedia's overall link structure.

4.3 Learning to Play Wikispeedia

Although the DS agent is much better than agents based on DFS or BFS and even beats the median human in many instances, its searches still degenerate into random walks on several missions (cf. plot $(1, 1)$ in Figure 4). This raises the question whether we can design an agent of more consistent performance, achieving human-like search times on all Wikispeedia missions. In this section, we present such an agent. It is based on the reinforcement learning technique of temporal difference (TD) learning, and we therefore call it TDspeedia.

4.3.1 The Learning Algorithm

Recall that the DS agent is simply a DFS agent with the only difference that it iterates over the current article's neighbors in order of increasing semantic distance to the goal of the current mission (cf. Section 4.1). The TDspeedia agent is yet another modified version of DFS. Its distinctive features are that

1. it iterates over the current article's neighbors in order of decreasing *value* (rather than increasing semantic distance, as in the DS agent) and

2. it learns a good function attributing values to states during game play.

Before describing in detail what the value function is and how the agent learns it, I will describe how the game of Wikispeedia can be modeled as a deterministic Markov decision process (MDP) in order to make it amenable to reinforcement learning tools.

A deterministic MDP is defined by a quadruple (S, A, T, R), where S and A are the state and action sets, respectively; $T : S \times A \to S$ is the state transition function, $T(s, a)$ being the state one reaches by performing action a in state s; and $R : S \to \mathbb{R}$ is the reward function, $R(s)$ being a numerical reward (or punishment) received for being in state s. Put sloppily, rewards tell the agent whether a state is, from a myopic viewpoint, good or bad: the higher the reward, the more desirable the state.

In the Wikispeedia MDP, states S are the Wikipedia articles, actions A are link clicks, $T(s, a)$ is the article connected to s via link a, and $R(s)$ is positive (we choose $+1$) if s is the goal of the current mission and a small negative number (we choose -0.001) otherwise. Our choice of rewards is motivated by the fact that we want the agent to eventually reach the goal state and to not wander around for too long before doing so. Note that the reward function R depends on the goal of the current mission.

An agent plays Wikispeedia simply by picking a link to click, out of all the links available on the current article. This choice can be encoded in a so-called *policy* function $\pi : S \to A$, which maps each article to the choice the agent will make in that article.

Given a policy π, we can define the notion of a value function $V_\pi : S \to \mathbb{R}$, which specifies for each state how much reward the agent will accumulate from that state onward if it adheres to policy π. More formally,

$$V_\pi(s_0) = \sum_{i=0}^{\infty} R(s_i),$$

where s_{i+1} is the article reached by clicking link $\pi(s_i)$ on article s_i.[5]

The goal of an intelligent agent, then, is to learn the optimal policy π^* that maximizes $V_{\pi^*}(s)$ for each state s (we write V^* for V_{π^*} from now on). The optimal policy need not be learned explicitly. It suffices to learn its value function V^*. By choosing in each state s the link that leads to the state s' that maximizes $V^*(s')$, the agent will follow π^* implicitly. This is the approach we take here.

Note that, since rewards depend on the goal of the current mission, so does the optimal value function. So the exact Wikispeedia value function is defined by $|S|^2 = 4{,}604^2 \approx 21$ million numbers. To learn this many parameters from experience is infeasible. We therefore strive to approximate $V^*(s)$ as a linear combination of features $\mathbf{f}(s)$ (a vector of scalar values) extracted from state s. In other words, we find weight parameters $\mathbf{w}$ that maximize

$$V_{\mathbf{w}}(s) = \mathbf{w}^{\mathrm{T}}\mathbf{f}(s)$$

for all states s.

Linear function approximation has the advantage that there is a simple incremental update equation that is guaranteed to converge to the optimal value function in the hypothesis space, which is not the case for more general parametrized representations [RN03]. Weights are adjusted according to the so-called temporal difference (TD) update equation. The agent starts with some initial weights $\mathbf{w}$, acting according to the induced value function $V_{\mathbf{w}}$. Upon observing a transition from state s to s', it updates $\mathbf{w}$ as follows:

$$\mathbf{w} \leftarrow \mathbf{w} + \alpha\,(R(s) + V_{\mathbf{w}}(s') - V_{\mathbf{w}}(s))\,\mathbf{f}(s).$$

We refer to [RN03, p. 778] for a derivation of this so-called 'delta rule'. May it suffice here to give an intuition: in the equation, $V_{\mathbf{w}}(s)$ is our estimate of the value of s before seeing the transition to s', and $R(s) + V_{\mathbf{w}}(s')$ is our estimate for it after receiving the reward $R(s)$, so the update equation adjusts the weights in a gradient-descent way, making the value function match our reward observations more closely. The parameter α is called the learning rate. We choose it to decrease over time: in the n-th training game it is set to $1/(1000 + 100n)$. This way there is more exploration of the weight space early on and less so later.

Algorithm 1 lists pseudocode for the resulting learning agent. As mentioned above, it implements a DFS that iterates over neighbors in order of decreasing value and updates the weights along the way. To learn the weights, we have the agent play a number of training games, where start and goal article are picked randomly. In game $n + 1$, the agent is started with the weights resulting from game n. To avoid overfitting, the weights are normalized after each training game such that the weight vector $\mathbf{w}$ has length 1 (this can be considered a simple form of regularization). After training, the weights are fixed and the agent uses the most recent weights in the test phase. In Algorithm 1, this simply corresponds to using a learning rate of $\alpha = 0$.

4.3.2 Features Used to Approximate the Value Function

Now that I have introduced the learning algorithm, it is time to discuss the features $\mathbf{f}(s)$ used in the value function $V_{\mathbf{w}}(s) = \mathbf{w}^{\mathrm{T}}\mathbf{f}(s)$. Since our agent should still perform a (more sophisticated) decentralized search, we may include only local features, i.e., information from the article to be evaluated (let us call it the candidate article) and the goal article. No global network information must be used. The PageRank of a canidate—or any other centrality scores, for that matter—is an example of a global feature.

We now list and motivate the features used in the current implementation.

[5] To deal with the infinite sum, we can define that, once the agent reaches the goal state g, it keeps looping in g forever, while receiving the positive reward $R(g)$ only once and zero reward thereafter.

Algorithm 1 TDspeedia

Input: start article s_0; goal article g; weights $\mathbf{w}$; learning rate α
Output: updated weights $\mathbf{w}$

```
push s_0 onto initially empty stack S
while stack S is not empty do
  pop article s from stack S
  if s = g then
    // final update: here the agent gets the positive reward R(g):
    w ← w + α (R(g) + V_w(g) − V_w(g)) f(g) = w + αR(g)f(g)
    return w
  else if s has links to articles N that have never been on the stack S then
    for s' ∈ N, in order of increasing V_w(s') do
      push s' onto S
    end for
    s' ← article that was last pushed onto S
    w ← w + α (R(s) + V_w(s') − V_w(s)) f(s)
  end if
end while
return 'failure'   // no path between s_0 and g exists
```

Semantic Relatedness Since the learning agent should be an extension of the DS agent, one feature should definitely be the one used by the latter: the semantic relatedness between the goal and candidate articles. Indeed, the DS agent is a special case of Algorithm 1, with the weight for this feature set to one and all others to zero, as well as a learning rate of $\alpha = 0$.

TF-IDF Cosine Another common measure of similarity between documents is the cosine between their so-called TF-IDF vectors [MRS08], a vectorial representation of their textual contents. The idea behind using several measures of semantic relatedness is to make the agent more robust: in case one measure does not capture the similarity between the candidate and the goal article, the other one might still be able to do so. To compute TF-IDF scores efficiently, I used the custom search engine Lucene [Apa09].

Outdegree If two candidate articles are roughly equally related to the goal article, then it is intuitively a wise choice to pick the article with more outgoing links, since it offers more options to continue the search. This is why we include the candidate's outdegree as a feature.

Phase It turns out that human Wikispeedia games can very often be divided into two phases: in the first, or getting-away phase, players try to reach a hub article with high outdegree that reaches into many different regions of the Wikipedia graph; in the second, or homing-in phase, the search narrows down by traversing articles that are ever more closely related to the goal semantically (cf. [WPP09] for more details). To give the agent a way of mimicking such behavior, we include what we call the Phase feature, defined as $d_s \log t$, where d_s is the outdegree of candidate s and t is the current time step. In accordance with the above, we expect the agent to learn a negative weight for this feature, such that articles with many outgoing links are more desirable at the start of a game, i.e., in the getting-away phase.

Immediate Success If the candidate article to be evaluated is itself the goal article, then trivially the agent should choose to go there. This feature is 1 if this is the case and 0 otherwise.

Nearly Immediate Success If the candidate article links to the goal article, then again the agent should go there, since success is nigh. This feature, too, is 1 if the candidate links to the goal and 0 otherwise.

Constant Bias We also include a bias feature that always takes on value 1, regardless of the state, as normally done in linear function approximation.

Several of these features live in disparate regions of the real number line: cosines take on a maximum value of 1, while outdegrees, e.g., can be two orders of magnitude larger. To give all features an equal

chance of having an impact, we normalize feature values to zero mean and unit variance, by subtracting the mean of the feature and dividing by its standard deviation (for the zero-variance Constant Bias feature, this is impossible, of course). Estimates for mean and standard deviation are obtained by collecting one million state transitions in 703 games of the basic DS agent.

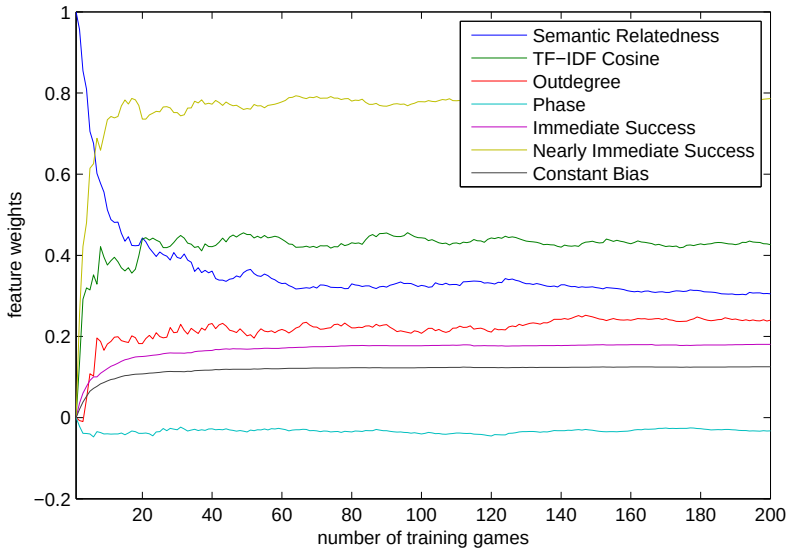

Figure 6: Change in feature weights as training progresses. In case no colors are available: after 100 training games (and thereafter) the curves represent the following features, from top to bottom: Nearly Immediate Success, TD-IDF Cosine, Semantic Relatedness, Outdegree, Immediate Success, Constant Bias, Phase.

At the beginning of training, the agent starts with only the Semantic Relatedness feature having weight 1 and all others having weight 0, i.e., it starts as a DS agent. Figure 6 illustrates how the feature weights evolve as the training phase progresses.

The Semantic Relatedness feature loses weight and is surpassed by the TF-IDF Cosine feature, which means that the latter is a better indicator of semantic relatedness in the Wikispeedia task.

As expected, the Outdegree feature gets a positive weight, i.e., in general, articles that offer a bigger choice of options are favored, and the negative weight of the Phase feature confirms our expectation that this is more so in the beginning of games than towards their end. Note that the strong Outdegree weight is also in tune with Milgram's original small world experiment [Mil67], in which it was found that many participants chose to route their letters through highly connected hub nodes.

The Nearly Immediate Success feature eventually attains the largest weight, which is reasonable, since it is a very strong indicator of the value of a candidate article.

At first it might seem surprising that its weight is even stronger than that of the Immediate Success feature. I conjecture that this is the case because the Immediate Success feature is highly correlated with the two semantic relatedness features (Semantic Relatedness and TD-IDF Cosine), which both take on their maximum values when the candidate article is the goal article, i.e., when the Immediate Success feature also fires. Note that the Immediate Success feature is useful despite this redundancy: in the absence of this feature the dominating weight of the Nearly Immediate Success feature might make the agent go to *yet another* candidate that also links to the goal rather than to the goal itself, although the two semantic relatedness features fire strongly in favor of the latter.

Note that the weights converge quickly. We in fact trained the agent for 1,000 games but show only the first 200, since barely anything changes after this. The weights are in their final order from around the twentieth training game onwards, and even their values do not change by much thereafter.

4.3.3 Performance of the Learning Agent

Let us now scrutinize whether learning from experience gives the agent an edge over the default DS agent.

A histogram of the search times achieved by the TDspeedia agent is shown in the top of Figure 7. For ease of reading, the figure also repeats the histogram of human and DS search times. The axes of all plots are scaled identically. TDspeedia's search times seem much more similar to humans' than the DS agent's do. The estimated power law exponent of $\alpha = 3.22$, e.g., is closer to the human $\alpha = 3.50$ than the DS agent's $\alpha = 2.79$. Indeed, the histogram appears shifted to the left, the mode now being at 3 rather than the 5 of humans.

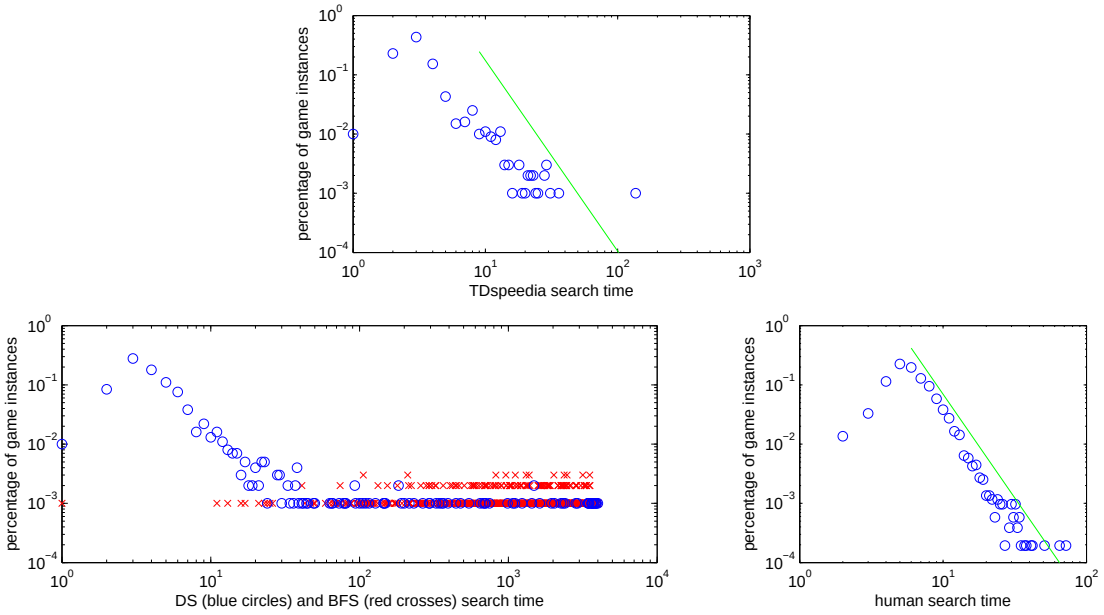

Figure 7: *Top:* Histogram of search times achieved by the TDspeedia agent to solve the 1,000 Wikispeedia test missions. The green line shows the estimated power law with parameters $\alpha = 3.22$ and $x_{\min} = 9$. *Bottom:* To minimize the amount of page-turning on the esteemed reader's behalf, I also reproduce Figures 3 (left) and 2, showing the search times of the DS agent and human players, respectively.

This is confirmed by the last row of the scatter plots in Figure 4. Plot $(4, 1)$ there suggests that the learning agent's performance is comparable to that of humans, while plot $(4, 2)$ shows that it significantly beats the DS agent. Whereas the latter degenerated into a random walker on numerous missions, this is no more so with the TDspeedia agent. In fact, plot $(4, 1)$ does not even properly emphasize the quality of the learning agent: as discussed in Section 4.1, each dot can represent several game instances. When tallying on how many of the 1,000 test missions the TDspeedia agent searches for no longer than the median human, one finds that this is the case for 893 missions. It is strictly faster than the median human on 818 missions.

One might argue that the comparison is slightly unfair, since the TDspeedia agent gets to look at all candidate articles linked from the current article for free, whereas explicitly looking at the candidate article would be counted as an extra click for humans. But then again, humans can use their *global* commonsense knowledge acquired in years of interacting with the real world to form expectations about what the candidate articles would look like, whilst the agent still uses only *local* information (even a one-step look-ahead is local), so I would rather make the point that the comparison does not favor the TDspeedia agent after all.

5 Perspectives

To summarize, the main insight gained while working on this project is that Wikipedia's hyperlink graph is efficiently navigable via decentralized search, for which I provide evidence by analyzing the human-computation game Wikispeedia, which asks players without knowledge about Wikipedia's global network structure to navigate between articles. I support my argument from several angles. First, humans manage to solve Wikispeedia missions successfully, rarely taking more than 10 clicks. A simple automated agent that always greedily chooses the link that minimize the semantic distance (according to an off-the-shelf computational semantic distance measure) to the goal article often performs competitively with humans. I then argue that the reason for Wikipedia's navigability is the fact that link lengths (measured in a rank-based fashion based on the underlying semantic distance measure) roughly follow a power law of exponent -1, as stipulated by Liben-Nowell *et al.*'s model [LNNK$^+$05]. While the simple decentralized-search agent still degenerates into a random walker on many missions, Wikipedia becomes fully navigable when the agent uses a richer distance function combining several features of the candidate articles with learned weights.

The success of the learning agent is encouraging beyond the small world that Wikipedia constitutes. A worthwhile avenue of future research might consist in learning to search in arbitrary graphs. The task could be to locate a specific piece of information in an information network, i.e., to find a path to an information source while exploring as few nodes as possible. Example tasks might be: 'Search an *a priori* unknown corpus of scientific papers to find the article containing a proof of Gödel's incompleteness theorems', or: 'Browse IMDb to find out which movie contains the Penis Song.'

Similar techniques might also be deployed in order to find information *similar* to a given document, rather than to locate the very document itself. For instance, given Gödel's paper 'Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme', search a large corpus for papers about similar topics, without having access to an index in which you can efficiently look up information about all documents in the collection. Instead, you can only follow up on references mentioned in the papers you encounter. This task could potentially be tackled by searching for Ü.f.u.S.d.P.M.u.v.S. itself. The documents found just before locating the target are then likely to be related to the latter.

Another application domain could be search in social networks. There as well, nodes typically have only limited local information, yet would like to navigate the network. For instance, an agent might want to become acquainted to a certain other person. The task would then be to get closer and closer to the target person, finally befriending people who can introduce the agent directly to the target.

I finally want to point out that, while the TDspeedia agent presented here is basically a DFS with a smarter way of pushing nodes onto the stack, there might be even cleverer variants of the search, more similar to A*. For instance, instead of always just popping from the top of the stack, one might as well pick the optimal node from amongst all the candidates that are currently on the stack (which then ceases to be a stack, of course). In certain tasks, such as Wikispeedia, this is admissible, while it is not legitimate in other types of networks, e.g., in physical networks, where routing corresponds to geographically navigating between neighboring nodes.

References

[Apa09] Apache. Lucene. Website, 2009. `http://lucene.apache.org` (accessed Feb. 9, 2010).

[CSN09] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009.

[Kle00] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *Proc. 32nd Annual ACM Symposium on Theory of Computing*, 2000.

[LNNK$^+$05] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *Proc. National Academy of Sciences of the United States of America (PNAS)*, 102(33):11623–11628, 2005.

[Mil67] S. Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

[MRS08] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[MW08] D. Milne and I. H. Witten. An effective, low-cost measure of semantic relatedness obtained from Wikipedia links. In *Proc. 1st AAAI Workshop on Wikipedia and Artificial Intelligence (WIKIAI'08)*, 2008.

[RN03] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[Wik07] Wikipedia. 2007 Wikipedia Selection for schools. Website, 2007. `http://schools-wikipedia.org` (accessed Aug. 3, 2008).

[WPP09] R. West, J. Pineau, and D. Precup. Wikispeedia: An online game for inferring semantic distances between concepts. In *Proc. 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, 2009.

P.S. The Penis Song is from Monty Python's 'The Meaning of Life', of course.