Human-Powered Word Sense Disambiguation

Nitin Seemakurty, Jon Chu, Luis von Ahn, Anthony Tomasic
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA
{nseemaku, jechu, biglou, tomasic}@andrew.cmu.edu

ABSTRACT
One formidable problem in language technology is disambiguating the true sense of a word as it occurs in a sentence (e.g., recognizing whether the word "bank" refers to a river bank or to a financial institution). This work explores a specific strategy for solving this problem. The strategy involves harnessing the linguistic abilities of human beings to develop datasets that can be used to train machine learning algorithms. Generation of quality datasets can greatly aid the development of algorithms that tackle challenges such as automated language translation and the development of a semantic web. To create such datasets, we introduce a new interactive system: a fun game designed to produce valuable output by engaging human players in what they perceive to be the casual task of guessing the same word as another player. Our system makes a valuable contribution by tackling a bottleneck in the WSD domain: knowledge acquisition. Rather than using conventional and costly techniques of paying workers to generate training data for machine learning algorithms, we delegate the work to people who are looking to be entertained.

Keywords
Word sense disambiguation, sense tagging, distributed knowledge acquisition, games with a purpose, online games

INTRODUCTION
The human language is ambiguous. That is, words can be interpreted with different meaning depending on their surrounding context. Take, for example, the following two sentences [7]:

(a) I can hear *bass* sounds.

(b) They like grilled *bass*.

The word *bass* refers to a low-frequency tone in one sentence while it refers to a type of fish in the other. Although this sense recognition seems intuitive to humans, it is a much more sophisticated task for a machine, which has to cope with the unstructured nature of the data (language). This computational identification a word's meaning in a given context is called *Word Sense Disambiguation (WSD)*.

The relevance of WSD is becoming clear as advancing information/web technologies are catalyzing the production of enormous amounts of textual data, including articles, blogs, status messages, digitized books, etc. There is a growing need to introduce structure to this data in order to make it consumable and manageable by machines. Unfortunately, WSD remains a difficult problem for several reasons.

Most relevant to our study is the issue of knowledge acquisition. At its core, a WSD system is one that utilizes available sources of knowledge to calculate the most relevant meaning of a given word in context. But manually creating a training dataset for a WSD system generally involves taking a large set of textual data, isolating words to disambiguate, and hand labeling each of these words with their "correct" meanings. We quickly find that this process is an arduous and consequently expensive one [6].

But what if we make this labeling process a pleasant one? This paper explores a new system, a game that is designed to capture human knowledge in a distributed fashion via an enjoyable game. Our study involves *assessing the effectiveness of this game in tackling the knowledge acquisition bottleneck*. Many elements of our system are derived from a predecessor: the ESP Game [10].

Open Mind Initiative
Like the ESP Game, our game is much in tune with the efforts of the Open Mind Initiative [9], which focuses on collecting data from internet users in order to train machine learning algorithms. Our game is similar in that it attempts to use the efforts of regular internet users to tag the senses of words. However, as with the ESP Game, we place particular emphasis on the playability (i.e. survivability) of our system.

GENERAL GAME PLAY
Jinx is an online two player game. When a player begins the game, he/she is paired with another random player. The player does not know who his/her partner is and the game does not facilitate any form of communication between the two players. Each player interacts with the game independently. The players share only one aspect of the game: the current round. At any given time, both players view the same round, where a round is defined by a context (e.g. a sentence), and a highlighted word within that context.

The players are encouraged to rapidly type replacement words/phrases for the highlighted term. They are given incentive to type words that their partner is likely to type because both players are awarded points if and only if they both type the same string. As with the ESP Game, these

players do not need to type their matching string at the exact same time, but both must have independently typed this string at some point during that round (See Figure 1 below).

Figure 1. Each player guesses word replacements independently. Neither player can see the other player's guesses.

We call this matching string a "tag". Once a tag is collected, the game awards points to each player and then proceeds to the next round. In the case where agreement cannot be reached, the round expires after 30 seconds. Players are presented rounds for exactly 3 minutes, and then they are taken to a summary page that recaps their performance and offers to restart the game.

Our observations currently indicate that these collected tags are typically appropriate synonyms for the highlighted term.

GAME DESIGN DETAILS

Our game was originally designed to be more of a quiz comprised of a series of multiple choice questions. The player would be presented with a highlighted word in context, and then given multiple definitions to choose from. These definitions were intended to reflect different interpretations of the highlighted term. The player would be rewarded if his/her choice matched with the partner's choice. This setup, however, was inherently flawed.

(a) **Random guessing.** This older setup allowed players to collect occasional points by blindly selecting answers and rapidly progressing through rounds, hoping for a lucky match with the partner. One solution could be to penalize for mismatch, but that would mean that a normal player would be deducted points due to the misbehavior of his/her partner. Our solution to this problem was inspired by looking at yet another flaw with this setup.

(b) **Rigidity.** Limiting answer choices to a set of dictionary definitions also had the potential to create confusion if none of the definitions "worked". Players would eventually choose an answer that only weakly approximates the word's true meaning. From our perspective, this translated to weaker data.

(c) **Playability.** Like the ESP Game, we diverge from the work of Open Mind by emphasizing the element of fun in human knowledge contribution. A game that demands a player to read through dictionary definitions and merely click through rounds would quickly be abandoned by the online community.

For these reasons, we adopted a more open-ended approach that allows players to rapidly guess word replacement strings. This approach minimizes a player's ability to match by randomly guessing. At the same time it makes the game more challenging and engaging by requiring that the players guess cooperatively, despite not being able to communicate. This cooperation emerges automatically from the task at hand and results in the generation of valuable tags.

Tag Quality

Because there are only a few word replacements that are relevant to any given round, players quickly recognize that making guesses from this limited set drastically increases their chances of matching with their partner. Consequently, the tags collected from the game are typically relevant word replacements.

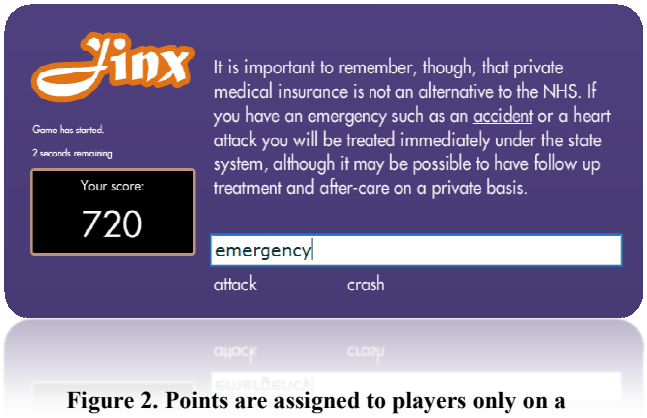

Figure 2. Points are assigned to players only on a match. The number of points rewarded depends on several different factors.

Point system

A game's reward system can drastically affect player behavior during game play. In designing the point system for Jinx, we had several goals in mind. We wanted to keep the game fast paced while still allowing for high quality input from players. Fast pace is encouraged because prior dry runs of the game indicate that the matching tag (i.e. the best replacement for the word) is commonly a very early guess during the round (one made quickly after reading the provided textual context). Giving players a sense of urgency encourages them to guess what is most intuitive to them, and this tends to be a successful tag. To generate this urgency, upon a matching guess, we reward each player $P = f(t)$ points, where t is the number of seconds remaining in the round when the matching guess was made, and f is an increasing function (we currently use $f(t) = 10 * t$). The faster a player generates a tag, the more points he/she makes. Notice that one of the players will inevitably

make the matching guess before the other player does; each is rewarded accordingly.

Players are also awarded bonus points for successfully matching with their partner on consecutive rounds. The value of the bonus increases on every round of their consecutive streak. At the end of a round, we award each player $g(n)$ bonus points, where n is the number of consecutive matches so far, and $g(n) = 10 * n$. Note that while g is a linear function, its effect is geometric because the total bonus a player receives is $10 * (1 + 2 + 3 + \cdots)$. This bonus system not only encourages players to keep playing, but also to keep playing with accuracy in mind. Most importantly, it keeps the game exciting.

Bots

During the course of a round, our game records all of a player's guesses for that given highlighted term in context, along with the guesses' timestamps. Whenever we need to deploy a bot, we intend to "replay" these guesses with the same delays as when originally recorded, as if they was coming from a human player. The partner to a bot should be completely unaware of its automation; all guessing and matching proceeds normally.

Because our game is a two player game, it is always possible for one player to have no one to pair with. In such a scenario, a bot will be equipped with a recording of a similar previous round and then paired with the solo player.

Bots will also be used to populate the game so that random matching can continue reliably. We discuss the importance of such matching in preventing cheating below.

Cheating

To maintain tag quality, it is imperative that paired players in our game are unaware of each other's identity. Otherwise, their guesses can easily be coordinated to match and then produce a bogus tag. While our system is designed to account for occasional bad tags, communication between two paired players can result in bad data.

It is easy for this communication to happen if both paired game instances are running at the same location. In such a case, there is only one player controlling both instances and it is trivial for that player to match on guesses. To minimize the chances of this one-player scenario, we record IP addresses and ensure that paired game instances run at separate IPs. It is also possible for a player to spawn multiple instances of the game from separate IPs in rapid succession, hoping that at least two instances under his/her control are paired. To mitigate the threat of this scenario, our server pairs all waiting players only once every thirty seconds (inspired by the ESP Game). This ensures (with high probability) that more than one player is in the waiting pool at the time of pairing.

Another threat to the game is mass-strategy. Mass-strategy refers to a consensus among a significantly large group of players to coordinate all their input (e.g. always make the same guess). The Internet makes the organization of such an attack very feasible. Inspired by the ESP Game, we aim to design our game such that when a mass-strategy is detected, a large number of bots can be immediately deployed. The pre-recorded game play of these robots severely limits the effectiveness of the mass-strategy. We presume that this bot infusion will be enough to render the mass-strategy a useless endeavor.

GAME EVALUATION

In evaluating our game, we looked to measure two things:

(a) **Correctness.** Does our game's output accurately identify the multiple meanings of a word? Is its output valuable to a machine learning algorithm?

(b) **Efficiency.** How fast does our game produce tags? How quickly does it create the data that is necessary to train machine learning algorithms?

To evaluate the game's correctness in collecting quality tags, we utilize data from the HECTOR project [1].

HECTOR provides us with a large set of contexts (usually complete sentences), each of which contains one demarcated word (; we will hereafter refer to such contexts as *challenges*. Each challenge is also assigned a definition ID, which corresponds to the particular definition that the challenge word carries in that particular context. Note that multiple challenges can contain the same challenge word and yet carry different definition ID's. This difference indicates that alternate meanings of the highlighted term are being invoked, where each of these meanings is given a separate ID.

To measure the game's ability to distinguish these alternate meanings, we injected a select subset of the HECTOR challenges into our backend and presented the game to a group of 11 players. The challenges were selected such that they involved only ten distinct challenge words but invoked multiple definitions of each of those words. There were no bots active during this trial run and it lasted one hour.

Trial Run Observations

Before we discuss correctness, we explore several observations we made after running our experiment:

1. The players' sentiment indicated that they found the game challenging, and consequently entertaining. While the players were keen to note several kinks in the implementation, most were intrigued by the concept. An account of the players' written feedback is presented in the appendix.

2. Players felt that the game was too time constrained. The time limit for each round was 30 seconds during this trial run. That is, players, once paired, had exactly 30 seconds to agree on a tag; otherwise, they were presented a new challenge. All participants in the study agreed, however, that 30 seconds was too little.

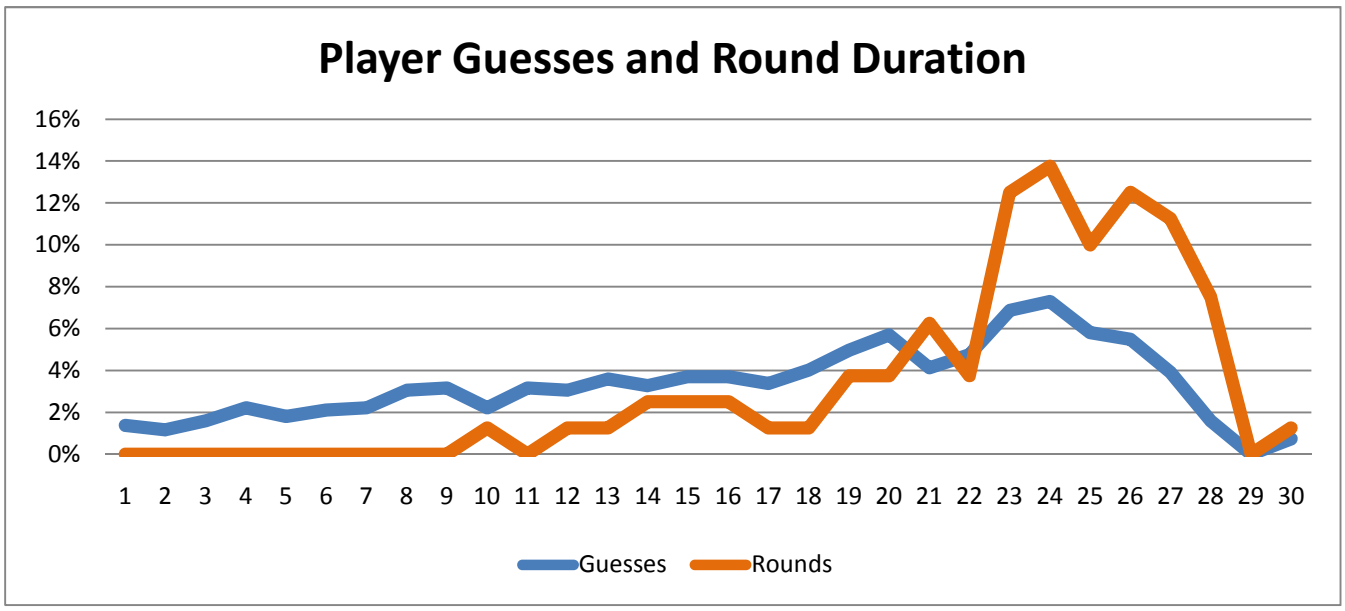

Figure 3. During our trial run, 46% of guesses happened between with after 20 seconds have passed on the timer. Furthermore, 83% of rounds take more than 20 seconds to complete. That is, 83% of the guesses that match happen after 20 seconds of player. ADD AXIS LABELS

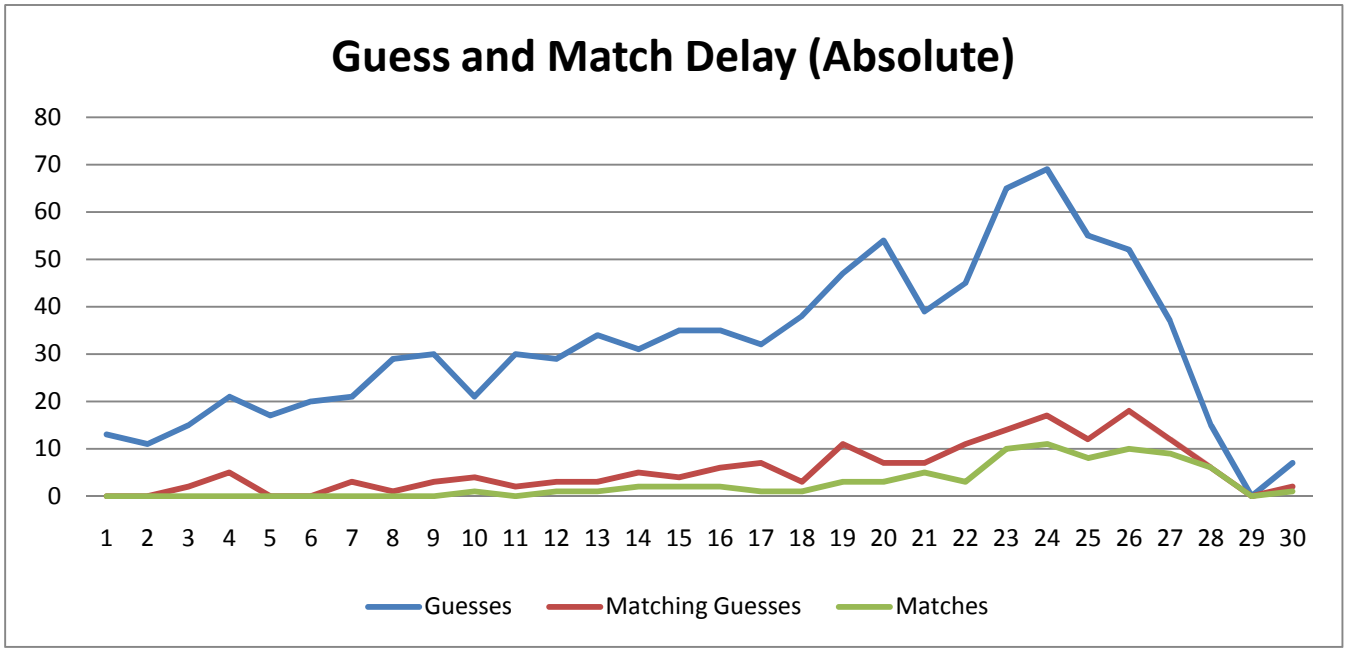

Figure 4. During our trial run, 46% of guesses happened between with after 20 seconds have passed on the timer. Furthermore, 83% of rounds take more than 20 seconds to complete. That is, 83% of the guesses that match happen after 20 seconds of player. ADD AXIS LABELS FIX COLORS

We attribute this time pressure problem to the relatively long contexts presented by HECTOR. It is common for a challenge to be 150-200 words long. Consequently it is difficult for players to read the sentence and be left with time to spare for guessing. Figure 3 plots the number of the guesses and successful matches that occurred at each timer tick. The plot indicates that the bulk of the user interaction with the game is happening within the last 10 seconds of each round. Although this behavior would exist to some extent regardless of the length of each round, the fact that the players are being rushed is being accentuated by how low the match rate is (compared to the dry runs we had on paper). The low match rate indicates either that the challenge word is difficult to replace, or that the proportion

of blind guessing is undesirably high. The problem here is that even penalizing for excessive guessing will not solve the problem, as this guessing largely a last-second scramble.

Interestingly, we were told by the players that, when presented an especially long challenge, they chose to read only the immediate context (5 to 10 words) surrounding the word. They communicated (and it is generally true) that much of the context provided by HECTOR is unnecessary to identify which meaning of a word is being invoked.

Although presenting too little context has a chance of heightening the challenge word's ambiguity, it is clear that presenting less context than HECTOR offers can not only make the game less time constrained, but also make the game appear less formal and more fun.

3. Players noted that there were some words that simply could not be replaced with a synonym. The word that troubled players most was blah. We know that this is not entirely avoidable. The only way to eliminate challenges with irreplaceable words is to allow humans to recognize this difficulty to begin with. Therefore, to alleviate this concern, we anticipate inserting into the game a "pass" button, which allows players to simply skip challenges that are too difficult. The problem with this approach is that the "skip" button might be too tempting. Disambiguation of words that are difficult to disambiguate is what makes the output of Jinx valuable. We do not want to make it too easy to skip these valuable challenges.

   One way to remedy this problem is to first detect "difficult" challenges by counting the number of times it is skipped. Then, instead of opening up the round to free guessing, we could present to them the set of definitions for the challenge word, and allow the players to match by choosing the same definition. This will ease the difficulty of coming up with a synonym by themselves, and at the same time add variety to the game. Points can be deducted on a mismatch to discourage random guessing, and mass strategy can be averted by randomizing the order in which the definitions appear on in the game.

Synsets

Before we discuss the results of our trial run, we introduce the reader to the concept of a synset.

We later intend to use Princeton University's WordNet [5] to map the tags we have collected to machine readable, standardized definitions. We plan to do this by utilizing the "synsets" provided by WordNet. Synsets are essentially a group of synonyms that, for the purposes of information retrieval, are semantically equivalent. WordNet attributes a simple but representative definition to each synset. By recognizing which WordNet synset most intersects the set of tags collected for a word, we can link that word to the synset's affiliated definition.

Trail Run Results

After examining the output of our trial run, it is clear that the algorithm satisfies our needs. That is, it successfully maps each definition of a word to different synsets. We have not had time to verify that these synsets overlap sufficiently with Wordnet synsets.

	A	B	C
1	Word	Definition	Guess
2	bitter	500637	bad
3	bitter	500637	bad
4	bother	502545	irritate
5	bother	502545	annoy
6	bothered	502545	annoyed
7	sacking	504753	firing
8	sacking	504753	firing
9	sacking	504753	firing
10	sacking	504753	firing
11	sack	504756	bag
12	sacks	504756	bags
13	sacks	504756	bags
14	sacks	504756	bags
15	sacks	504756	bags
16	sacks	504756	bags
17	sacks	504756	bags
18	calculating	510255	clever
19	calculate	510343	find
20	modest	510839	poor
21	modest	510839	small
22	modest	510839	small
23	invade	511340	trespass
24	invade	511340	attack
25	invade	511340	attack
26	invade	511340	attack
27	invaded	511340	attacked
28	invaded	511340	attacked
29	invaded	511340	stormed
30	generous	512274	giving
31	generous	512310	large
32	Generous	512310	large

33	generous	512310	large
34	Generous	512310	large
35	excess	512405	extra
36	excess	512405	extra
37	excess	512405	extra
38	excess	512405	extra
39	excess	512472	extra
40	excess	512472	extra
41	behaviour	512740	action
42	behaviour	512740	action
43	behaviour	512740	action
44	behaviour	512740	action
45	behaviour	512740	actions
46	behaviour	512741	action
47	behaviour	512741	actions
48	behaviour	512741	actions
49	shake	516365	nod
50	shakes	516520	smoothies
51	knee	516619	leg
52	knee	516619	leg
53	knee	516619	leg
54	kneed	516648	kicked
55	kneeing	516648	kicking
56	giant	530184	huge
57	giant	530184	large
58	giant	530184	big
59	giant	530184	big
60	giant	530184	big
61	accident	532674	mistake
62	accident	532674	mistake
63	accident	532674	mistake
64	accident	532674	incident
65	accident	532674	mistake
66	accident	532674	incident
67	accident	532674	incident
68	accident	532674	chance
69	band	532806	group
70	band	532806	orchestra
71	slight	537043	small
72	slight	537043	small
73	slight	537043	small
74	slight	537043	small
75	slight	537043	small
76	slight	537047	attack
77	slight	537047	attack
78	promise	537626	good
79	brilliant	538322	smart
80	brilliant	538322	genius
81	brilliant	538322	genius
82	brilliant	538322	smart
83	brilliant	538324	shiny
84	brilliant	538324	bright

REMOVE SOME ROWS OF THE TABLE

ADJUST HIGHLIGHTS IN TABLE

Each row in the table above represents one match that occurred between two players during our trial run. The first column represents which challenge word they tagged, the second column represents the challenges HECTOR definition ID, and the last column shows what tag the players assigned to the challenge word.

The key property to notice here is that in most cases, when more than one definition had been matched on, the tag that was generated fairly reflects the difference in definition for that word. For example, take the challenge word “brilliant” at the end of the table. There are two definitions that were tagged by players, 538322 and 538324. For the 538322 definition, the tags revolve around intellect, whereas for the 538324 definition, the synonyms revolve around appearance.

INSERT SAMPLE SENTENCES THAT USE EITHER VERSION OF BRILLIANT

We noticed during our analysis that looking at only the tags generated by players gives us a much higher quality output. As anticipated, looking at all the guesses made for a given challenge does not give us much useful information. For example, “bright” was used as a guess on both versions of brilliant, but this is because “bright” itself has multiple definitions, at least one for both versions of “brilliant”. This, though, is only a slight problem because at scale, more than one tag will likely be generated for each version of a challenge word.

Note that not all tags are valid. If we look at the tags collected for “modest”, we notice that even though only one definition of it was recognized (510839), it was assigned two tags that do not align. We feel that this problem would also become minor at scale, as synonyms that mean “small” would eventually outnumber synonyms that resemble mean “poor”.

Otherwise, when multiple definitions of a word were not tagged, the one definition that WAS tagged, seemed to be tagged with consistent synonyms. In the table above, no challenge word was given two tags from the same pair of players. This means that the repeat tags we observe for many challenge words are truly agreed to be representative synonyms of the word in context. This allows us to take that word to WordNet and confidently say that whichever synset that tag appears in corresponds to the correct definition.

In the case where we only collect one tag for a word, and find that that tag appears in multiple synsets of WordNet, then we have a problem. At this point, we would require some sort of taboo system in the game, to encourage users to produce tags that have not already been collected. Or, we could use the "show them the definitions" approach mentioned earlier.

FUTURE WORK

Although the game is giving us satisfying results, we have not yet gotten far enough to verify that the game is producing output that we can manageably feed to a word sense disambiguation algorithm. The fact that the game disambiguates is promising, but whether or not the output format is useful for a machine learning algorithm, we do not know. For this we require much more data.

CONCLUSION

Ultimately, we hope to build a high quality dataset comprised of the following mapping:

$$(word, context) \rightarrow definition$$

We aim to do this in a distributed fashion by employing human beings to take a word in context and generate its tags, tags which later serve as links to the word's correct definition. The total mapping we generate can then be used as training data for machine learning algorithms.

To facilitate this process, we introduce our new system (Jinx), which is designed to effectively harness the linguistic abilities of human beings so that this mapping can be generated with quality and cost-effectiveness.

We hope that the data collected from Jinx can significantly lower the barrier to entry in WSD study and expedite, if not enable, the development of novel algorithms for language technologies.

REFERENCES

1. A. Kilgarriff. 1998. Senseval: An exercise in evaluating word sense disambiguation programs.
2. Anderson, R.E. Social impacts of computing: Codes of professional ethics. *Social Science Computing Review 10*, 2 (1992), 453-469.
3. How to Classify Works Using ACM's Computing Classification System. http://www.acm.org/class/how_to_use.html.
4. Mather, B.D. Making up titles for conference papers. *Ext. Abstracts CHI 2000*, ACM Press (2000), 1-2.
5. Miller, G. A. 1995. WORDNET: A Lexical Database for English. Communications of ACM
6. NG, T. H. 1997. Getting serious about word sense disambiguation. In Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How? (Washington D.C.). 1–7.
7. Roberto Navigli. 2009. Word sense disambiguation: a survey. ACM Computing Surveys, 41
8. Schwartz, M. *Guidelines for Bias-Free Writing*. Indiana University Press, Bloomington, IN, USA, 1995.
9. Stork, D. G. The Open Mind Initiative. IEEE Intelligent Systems & Their Applications, 14-3, 1999, pages 19-20.
10. Von Ahn, L. and Dabbish, L. 2004. Labeling images with a computer game. In Proc. ACM CHI.NG 1997 Paper
11. Zellweger, P.T., Bouvin, N.O., Jehøj, H., and Mackinlay, J.D. Fluid Annotations in an Open World. *Proc. Hypertext 2001*, ACM Press (2001), 9-18.