UNIVERSITY OF ALASKA, ANCHORAGE

CSCE A470

CAPSTONE PROJECT

A novel probabilistic language-based CAPTCHA system

Author:
Teslin ROYS

Supervisor:
Dr. Saif AL ZAHIR

March 2015

Computer Science & Engineering Department
UNIVERSITY *of* ALASKA ANCHORAGE

Acknowledgements

Many thanks in particular to my supervisor, Dr. Saif al Zahir for his persistent help in motivating and focussing the work. I would also like to thank all my family and friends for dinner, music, conversation and support which is ever appreciated when the work is trying. Since I would rather not forget anyone in these acknowledgements I won't list anyone by name, but if you ever gave me any kind of encouragement (whether you knew it or not) then thank you.

Contents

List of Figures

List of Tables

Abstract

In this paper, we argue that the notion of a CAPTCHA (Completely Automated Public Turing Test to tell Computers and Humans Apart) system which sorts humans from machines with perfect accuracy is not merely technically challenging but tightly connected to the philosophy of AI and therefore fraught. We embrace Von Ahn et. al. [2003]'s suggestion that new CAPTCHA designs be based upon difficult artificial intelligence problems, but we further recommend that new designs target areas of lesser development absent the expectation that an optimal solution will be found. CAPTCHA systems are typically primarily visual, characteristically challenging a user to identify a string of alphanumeric characters in an image. Many CAPTCHA systems in everyday practical use specifically employ the optical text recognition (finding characters in an image) approach, but we observe that this approach is not robust against increasingly well-developed attacks in the literature. We review newer methods, based on either image recognition or language semantics. We design a new CAPTCHA system in the latter category, but unlike existing examples of either type our system is not based upon a static, manually collected database and is not susceptible to rudimentary random guess attacks. Preliminary testing shows near-perfect acceptance of human users by the system, where comparable alternatives in the image recognition family have more inconsistent results. Using a dynamic social feedback mechanism where users score new phrases, this system is capable of identifying new synonyms and could also offer a measure of quality for automatically generated text.

Keywords: CAPTCHA, semantics, syntax, language, AI, philosophy

1 Introduction

1.1 Motivation

A CAPTCHA, sometimes called a Human Interactive Proof, is a variation of the Turing test (Turing [1950]) in which a system authenticates users as human by verifying solutions to a problem which is in principle straightforward for humans to solve but difficult for current computers or algorithms. In general, preventing undesired (automated) use of Web services is an apparent security need which CAPTCHAs attempt to satisfy. Mehra et al. [2011] argues that CAPTCHAs also present a viable means of mitigating the damage done by DoS (Denial of Service) attacks. There seems to be significant benefit in introducing new CAPTCHA techniques. As Von Ahn et al. [2003] suggest, the introduction of challenging AI problems for CAPTCHA mechanisms either leads to more secure systems or encourages advances in the field of AI. In addition, we observe there is an advantage in leveraging problem-solving abilities of human users to generate external benefits. For example, the reCAPTCHA system described by Von Ahn et. al [2008] uses results from users to help solve hard character recognition problems beyond the abilities of present algorithms.

A large family of CAPTCHA methods are characterized by taking a string of characters, visually transforming/distorting it, and asking the user to respond with the original string. Lupkowski et. al [2008] calls these optical character recognition (OCR) based methods, but we prefer the term optical text recognition. This is because as Chellapilla et. al. [2004] note, for the single-character problem, computers in fact have better results than humans. Systems in this family instead often rely on the difficulty of segmenting an image that represents a longer piece of text into its constituent characters.

However, many of the most-used optical text recognition based systems have been broken (Baecher et. al. [2011], Yan et. al. [2007], Yan et. al. [2008], Moy et. al. [2004], Mori et. al. [2003], Chellapilla et. al. [2004]). For example, Baecher et. al have found means to subvert versions of the reCAPTCHA system, which Baecher et. al still consider comparatively robust, with a success rate as high as 12.7%. This also indicates that new approaches are desirable.

1.2 Organization

The remainder of this paper will discuss three main topics. In chapter 2, we discuss the aims of a CAPTCHA system in general. In chapter 3, we present a review of a selection CAPTCHA approaches. In chapters 4-7, we discuss the proposed new probabilistic language-based CAPTCHA system. This system promises to provide some benefits to natural language processing in addition to offering a practical CAPTCHA solution. We will discuss some of the tradeoffs of the proposed method and an overview of its design.

2 Aims of a CAPTCHA system

2.1 Security features

It seems that a CAPTCHA system should ideally be efficient with respect to the time spent to generate a problem relative to the time spent by an attacker to solve it. In other words, there must seemingly be an asymmetry in the capability of algorithms to generate problem cases with solutions, and the capability of algorithms to solve an arbitrary case. This would appear particularly true when the goal is to stymie DoS-style (Denial of Service) abuse, as Mehra et al. [2011], because the aim may be to impose a computational cost on the attacker.

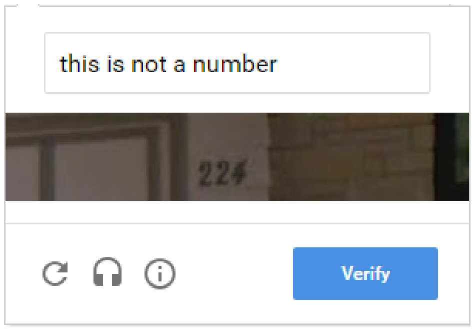


Figure 1: A challenge generated by the reCAPTCHA system, described in Von Ahn et. al. [2008]. The user must type in the characters (i.e. 224) they see in the image to proceed.

In addition, there must be an asymmetry between the algorithmic solver and a human user's ability to concoct a solution. This is apparently the case with OCR (optical character recognition) based systems, where the distorting transformations of typical CAPTCHA methods are more difficult to arbitrarily reverse than to apply. Nonetheless, it seems as though humans can find such a reversal trivial.

2.2 Usability

In order to gain any traction as a practical solution, we must expect that a provided CAPTCHA problem be not only less difficult for a human to solve than a computer, but require little time or effort on the part of the user. In addition, we would hope that the system's interface is readily accessible and understandable by most human users. A balance must therefore be struck between complication of the scheme in order to befuddle machines, and streamlining to ensure ease of use by humans. The hope, of course, is that these are at least somewhat orthogonal goals.

2.3 Philosophy

It further reflects on our goals for any CAPTCHA system to hypothesize about a CAPTCHA which could sort humans and machines with perfect certainty. If machines and humans have fundamental and irreconcilable differences in the tasks and problems that they are able to solve, then research should be directed at identifying these fundamental differences and exploiting them.

It is therefore reasonable to make a foray into the philosophical to ask whether one can set such a task which a machine finds not only difficult, but impossible, so that a machine will never be able to compose an answer. Of course, even hypothetically we are not simply defining a completely impossible task – to the contrary, a human must still be able to accomplish it. To suppose such a task to exist, we must suppose some unique and qualitative (rather than quantitative) difference between human and machine intelligence in principle. In other words, it posits a semantic gap (in the sense Von Ahn et al. [2003] describe it) which is in fact not incremental in nature but dramatically abrupt.

The existence of this kind of inherent qualitative difference between mind and machine is proposed by multiple authors including Penrose [1999] and Searle [1980]. The most resilient such claims frequently entail an appeal to the problem of syntax versus semantics. In some way, an objection to the principle of machine intelligence is usually offered on the grounds that a machine cannot perceive semantics. But a hidden premise is also that the distinction between syntax and semantics is rigorous enough to allow us to draw such a line.

Penrose's argument is particularly interesting from a computational logic point of view, deriving from the idea of Gödel sentences. Gödel [1931] showed that given an axiomatic language of sufficient complexity, there will be true sentences in the language which cannot be proven. It is known from Turing [1936] that equivalently some problems do not have a decision procedure (undecidable problems). It is from our seeming ability to construct these Gödel sentences (or similarly, concoct undecidable problems) that Penrose argues we humans have special capabilities that machines do not.

However, this argument alone seems less than convincing. As McCarthy [1990] points out, it is possible to create a program in LISP which can construct Gödel sentences of its own once provided with some confidence independent of the formal derivation system. The question, then, is where human confidence in semantic truth comes from and if it is fully transferable or replicable. Presumably, if a machine were supplied with its own such confidence, it could pass Penrose's bar.

Other arguments against the possibility of human-indistinguishable machine intelligence run along similar syntax versus semantics lines. For example, Putnam [1980] makes the case that the Löwenheim-Skolem theorem of set theory can be applied to show how syntax underdetermines semantics. But even this is, again, a system-dependent argument. As Putnam says, the Löwenheim-Skolem theorem demands the Zermelo-Fraenkel (ZF) axioms to hold, not to mention the Axiom of Choice (ZFC). This is debateably using insufficiently fundamental tools to answer a more fundamental question.

Regardless, if reasonable doubt exists whether there is an absolute qualitative distinction between minds and machines, it supports a theory that CAPTCHA research should aim to emphasize finding relative improvement in the success rates of humans over machines. Not only may a perfectionist CAPTCHA be practically infeasible, it is also unclear if there is theoretical basis for its existence. Areas where the semantic gap is wide and appears to be narrowing slowest are therefore the most promising for developing new approaches. One area in which the gap between machine and human capability is known to be very wide is understanding natural language. It is unknown precisely how difficult understanding natural language is, but as Chomsky [1957] shows, we know that English (for example) is not a regular language (readable by computer of limited computational capability, a deterministic finite-state automaton). That natural languages like English have still has not been classified further in the Chomsky hierarchy of languages (e.g. context-free, context-sensitive, or recursively enumerable) is evidence of the difficulty of the understanding or processing natural languages by machine.

3 Overview of existing approaches

3.1 Optical text recognition CAPTCHAs

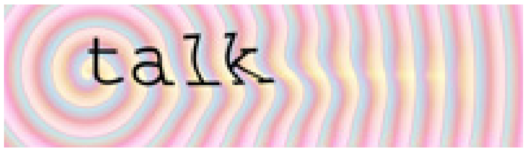


Figure 2: An example of an EZ-Gimpy image. Photo courtesy of the Captcha Project at Carnegie Mellon University.

Perhaps the most familiar type of CAPTCHA system, these exploit the gap in optical text recognition between machines and humans. Typically, the user is asked to read a short (frequently randomly generated) string of text which is visually distorted with noise, rotation, and various other transformations. These schemes can be referred to as text-based, but they should be confused with systems which primarily exploit a gap in comprehension of language features. As Baird and Riopka [2005] notes, a common attack on these type of schemes is based upon segmenting and identifying individual characters in the text.

Mori and Malik [2003] indicate that the popular Gimpy and EZ-Gimpy visual CAPTCHA implementations admit an automated solution in excess of 33% of the time, and advise that this makes the mechanisms an unreliable defense. However, as discussed above, it is possible that the effectiveness of an implementation might be better measured by the computational cost imposed on an attacker rather than its ability to thwart all such attacks. Nonetheless, this is certainly an indication that there are improvements to be made over the strategy employed by these systems.

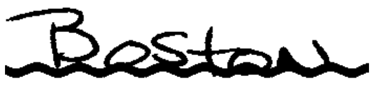
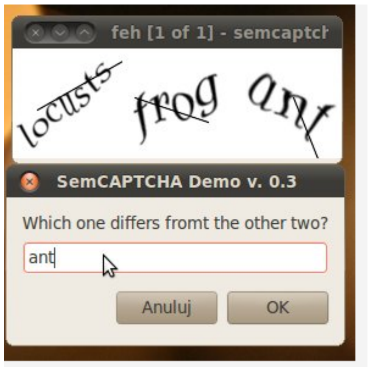


Figure 3: An example of a simulated handwritten test. Rusu and Govindaraju [2004]

The handwriting based system Rusu and Govindaraju [2004] propose is perhaps such an improvement. Rusu and Govindaraju [2004] argue that natural handwritten text offers a greater challenge for word/letter segmentation and character recognition than conventional automated distortions and filters (e.g. like those used by Gimpy).

3.2 Language-based or mixed approaches

A wide variety of alternate methods have been proposed. Some, like the purely semantic and language-based EGGLUE system considered by Hernandez-Castro et al. are implemented by a sizeable number of websites. The EGGLUE system has a couple of notable usability advantages. Namely, it is better accessible for blind or otherwise disabled users via publically available text-to-speech software than typical optical text recognition based systems, and the constraints on the answer (must be an English verb) make it potentially less frustrating to pass for the average reasonably literate English language user.



SemCAPTCHA Demo v. 0.3

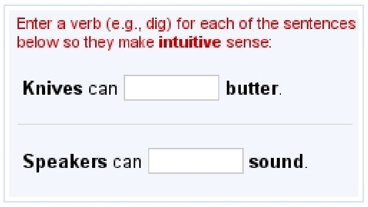
Which one differs fromt the other two?

ant

Figure 4: A sample prompt/answer for the SemCAPTCHA system. (Łupkowski and Urbanski [2008])

Hernandez-Castro et. al give a good account of the faults of the EGGLUE implementation, including a reliance on common verbs and a general susceptibility to search-engine based attacks. The system appears to rely on a static teleological database and the solution always consists of a verb, which makes it particularly vulnerable to a dictionary attack. Though the authors make a very good case about the flaws of the implementation, their broader conclusions about the viability of logic- or semantics-based CAPTCHAs and their recommendations regarding retreat from this general research direction seem less well founded. A general appeal is made to improving data mining techniques rendering semantics-based systems easily bypassed. But without a more substantial argument, the counterexample with regard to optical CAPTCHAs of steadily improving OCR techniques is easily deployed.

Systems can employ a multi-modal approach, as Almazyad et al. [2011] suggests, relying on a combination of simultaneous CAPTCHA mechanisms. This seems reasonable, as an attacker must then uniquely develop a solver which addresses a potentially unique combination of layered mechanisms. In general, we might tend to regard multi-factor authentication as being more secure. For this reason, ATM cards typically combine a token with a secret (PIN).


Enter a verb (e.g., dig) for each of the sentences below so they make **intuitive** sense:

Knives can ______ **butter**.

Speakers can ______ **sound**.

Figure 5: A prompt for the EGGLUE system.

SemCAPTCHA (Łupkowski and Urbanski [2008]) is an example of a multi-modal CAPTCHA which uses two techniques: (1) an ”odd one out” approach where several possible Polish word answers are given and a semantic connection exists between all but one answer, and (2) the text of these answers are visually distorted. Notably, the example implementation of SemCAPTCHA relies on a small and static semantic database based on animal taxonomy (mammals, reptiles, etc.) which, if known or guessed by an attacker, would make the system vulnerable. The “odd one out” approach makes the system vulnerable to blind (random guess) attacks, if the text recognition portion of the task is completed.

However, visually distorting transformations are not the only additional technique that could be added to confuse automated readers. Speaking even only of language-based approaches, there are a number of potential methods which may be examined. Misspelling, in-sentence word jumbling and abbreviation are a few kinds of basic transformations language-based CAPTCHAs might use to obscure words from identification by machine. A word may be misspelled by randomly altering a number of its letters to form a different but alphabetically similar word, and a human reader will usually nonetheless be able to correctly identify the word from context. Likewise, even more insidiously, a few words may be swapped for other genuine words (from a dictionary) – this is called a real-word misspelling. Correcting for real-word misspellings is known to be a challenging problem, though progress is being made with efforts like Wilcox-O'Hearn et. al. using trigrams. Words may also be rearranged in a sentence without unduly harming a human's ability to correctly determine its semantics (out of a list of grammatical sentences a human might identify the one with the most coherent meaning regardless of word order). Abbreviation or shorthand is another method of obscuration,

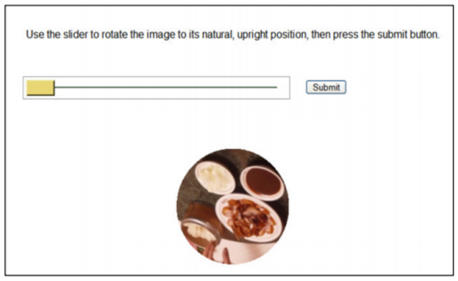

Use the slider to rotate the image to its natural, upright position, then press the submit button.

Figure 6: A sample prompt for the What's Up CAPTCHA system.

perhaps especially interesting because it seems to employs a surprising shift of interpretation for a machine. For example, the phrase “I see you” can be abbreviated to the phrase “i c u” by phonetic analogy. In general, natural language is challenging to analyse.

3.3 Other approaches

Gossweiler et al. [2009] offers a computer vision based system which does not depend on OCR. This system asks the user to submit a correct orientation (e.g. what is ”up”) for a displayed image. Photographs are largely assumed to be taken in an upright position, but the authors do not entirely rely upon this, as they expect their social feedback mechanism to correct for these kind of differences. Images of easily-detected objects are pruned from the database based on success and prevalence of specific object detection algorithms (e.g. images containing faces are omitted from selection), and images which are difficult for humans to orient are also removed via the social feedback mechanism. They identify images which are difficult for humans to orient upright by evaluating the average and deviation in angles that multiple users submit for a single candidate image. The presumption is that images with high variation in submitted angle are difficult to orient due to the lack of consensus.

Ross et al. [2010] followed Gossweiler et al.’s approach, but used line sketch renderings of 3D models instead of photographs. This has the advantage of providing a means to generate many distinct images (by viewing the model at different angles) in an automated fashion. Consequently a smaller database must be maintained to produce the same variety of CAPTCHA problems.

4 Description of proposed method

4.1 Principle

Inspired by recent work in developing website navigation techniques by Dhu et. al using a new method called 3-pole wafer trait sorting, this CAPTCHA system would sort potential users as human or machine by presenting them with three phrase options (3 poles) and allowing them to give a 'fuzzy' (probabilistic) answer about which phrase was human-written.

As is typical, the server authenticates users as human by issuing problems to solve at the request of the client. In this case, the user weights the three phrases with the probability they believe each phrase was written by a human, and submits the weights as a solution. The server evaluates the solution by degree of quality/correctness by dynamically comparing it to the solutions of other (passing) users and an initial database of basis phrases. If the user solves sufficiently many (ideally no more than a few) problems successfully they pass the test and are allowed access to the web service.

4.2 Choosing basis (match) phrases

Princeton University's WordNet (Miller [1995]) is a lexical database which stores extensive relational semantic information. The database encodes words as synonym-sets (or synsets) or groups of words with identical meaning. More information such as hyperonymy (super-subordinate relations) is also included, and an additional WordNet database stores teleological (purpose) information about words. For example, an audience is a (typical) "beneficiary" of showing a movie, which is encoded with the beneficiary relation.

There are many ways to retrieve or construct meaningful phrases from WordNet. Using the semantic relationships given between words, it would be possible to construct simple sentences, for example ("a simpleton is childlike" could be found by following semantic links in the portion of the database shown in Fig. 4.1). Instead, for a proof-of-concept of the design, it suits our purposes to take advantage of the various usage examples (usually consisting of a simple phrase) stored for each synset in WordNet. For example, one usage example for the word "border" might be "the soldiers marched across the border".

We select phrases stochastically by picking a random word in the database (i.e. synset element) and searching the database for a corresponding synset entry. From the synset we may extract the database's usage example sentences which meet certain parameters (e.g. length or presence of certain parts of speech). Having found a sufficiently large set of such phrases, we

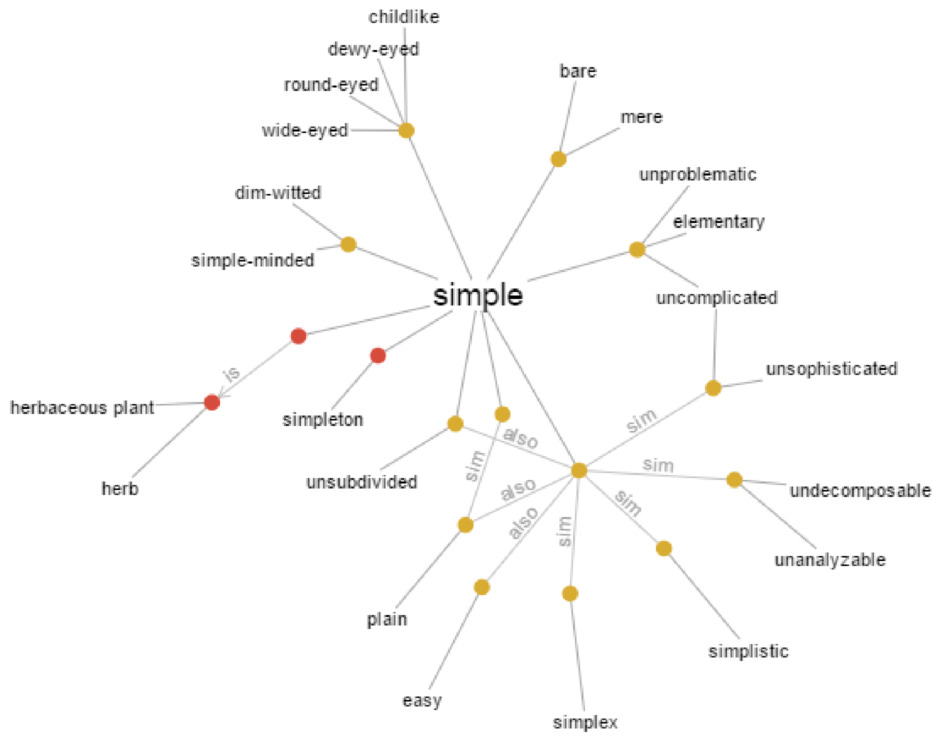


Figure 7: A visualisation of a small part of the WordNet database (Vercruysse [2010]). Words connected directly to the same node in the graph share a definition (i.e. compose a synset). Red nodes identify nouns, orange nodes are verbs.

may employ this as the basis of our pool of ”match” phrases, discussed further below.

4.3 Problem structure

A problem is presented to the user to identify which of the three phrases provided are most meaningful. The user is provided with three such phrases, with one being a phrase known to be meaningful and the other two being randomized or mutated examples. A problem, then, is a 3-tuple (M, C, R) consisting of the three phrases, with each phrase being either randomly generated, mutated, provided or known.

In the scheme, one of the three example phrases will be the known meaningful match M. One of the unknown examples will be a generated candidate C. The other, R, will be randomized and presumed incorrect until the user is already authenticated as human. M and C are randomly selected from a pool of match and candidate phrases respectively.

This simple problem structure is presented with out of a desire to keep the proof-of-concept system minimalist rather than because of methodological constraints. It is not difficult to envision a more sophisticated implementation which might add more aspects to the problem, such as employing visual distortion on the text offered to the user to provide additional challenge to attackers.

4.4 User choice mechanism

A user is presented with a triangular area with each vertex of the triangle labeled with one of three phrases. The user is prompted to click the cursor within the triangle nearest the one or two phrases which were 'most likely written by a human being'. A probability score is displayed next to each vertex M,C, or R based on where the cursor is pointed at coordinates K. The probability score S_i for a given phrase is the area of the opposite triangle made by the three other points in the figure, proportional to the size of the

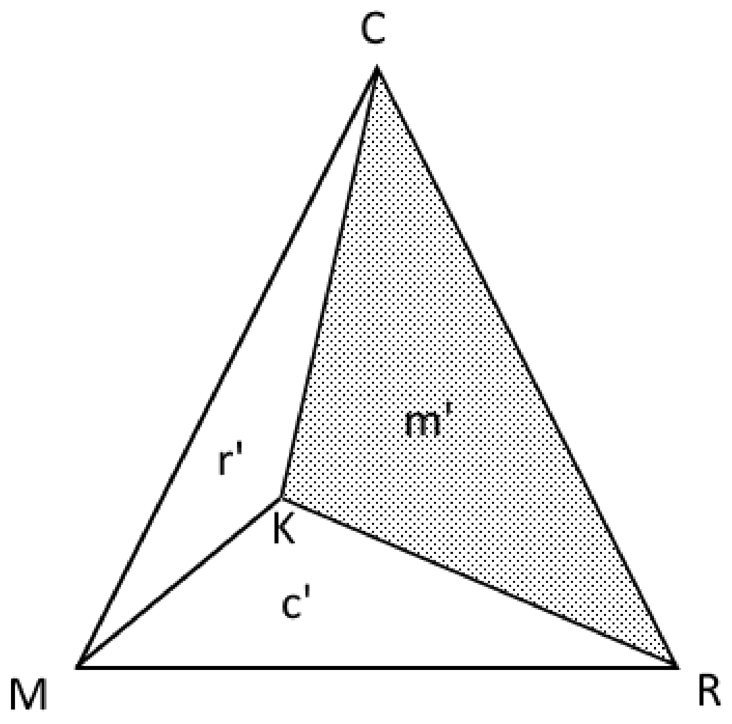


Figure 8: Highlighted region m' is the area $[CKR]$.

exterior triangle. See Figure 8. Here, the probability of M is $S = \frac{[CKR]}{[MCR]}$, where the area of opposite triangle CKR (highlighted region m') is divided by the area of overall triangle. A user's overall choice can be represented as a 3-tuple (m, c, r) of certainty scores taken for each phrase (vertex of the triangle).

threshold	purpose
T_1	Correctness (accepting users)
T_2	Incorrectness (rejecting users)
T_3	Candidate phrase suitability (condition for promotion)
T_4	Candidate phrase unsuitability (condition for dropping anomalies)
T_5	Random phrase suitability (condition for promotion)
T_6	Minimum number of users (condition for promotion)
T_7	Second thought threshold (condition for demoting a match back to candidate status)

Table 1: Threshold values for the system.

4.5 Passing/failing and multiple attempts

The user's choice is given in terms of degree of certainty, so too should the acceptance/non-acceptance be based upon a threshold of certainty as well.

The 3-tuple (m, c, r) of the user choice is examined with reference to the matched phrase, the candidate phrase, and the random phrase. Let $Q_1, Q_2, \ldots, Q_n$ be the quality rating of a sequence of successive choices by the user. The degree of quality of the choice numbered i is given by the following:

$$Q_i = e^{m_i} - e^{10r_i}$$

If the $\sum_i^n Q_i$ exceeds some positive threshold T_1 then the user passes. An insufficiently high Q can result in one of two consequences. First, the user may be given another problem case to solve if $\sum_i^n Q_i$ is positive. Second, the user may be rejected if $\sum_i^n Q_i$ negatively exceeds some negative threshold T_2.

number	phrase	scorers	$\sum$scores	average
p_1	"terrible luck this time"	14	5.103	0.364
p_2	"the mean annual rainfall"	7	4.58	0.654
...	...	...	...	...
p_n	"can't get no satisfaction"	11	7.81	0.71

Table 2: Example data for a candidate phrase database of n phrases.

4.6 Candidate formation

Candidate phrases can be formed in one of several ways to ensure diversity. Firstly, a random phrase may be promoted to the candidate pool if any single passing user has certainty r exceed some threshold T_5.

Secondly, both to introduce diversity into the system and to help identify synonyms, match phrases are mutated by a random swap of one or more words in the phrase taking on the same part of speech as the original word. This mutated phrase can then be added to the candidate pool.

Finally, it is possible to provide candidate phrases from some external source – for example, a text summarization algorithm. The success of candidates from that source in being promoted might then be an indicator of the quality of the text summarization algorithm.

4.7 Feedback mechanism

Feedback is considered in two cases: (1) for candidate phrases and (2) match phrases. First, a candidate phrase can be promoted to a match phrase if a consensus of passing users forms. The promotion of a candidate phrase C to a match can only occur if a sufficiently large number of (passing) users have evaluated it. Promoting a candidate phrase is then contingent on the average of the certainty scores A_c of the k passing users who considered the phrase C in some problem (M, C, R). In the following, c_j is the certainty score of user j normalized to a value between -1 and 1:

$$A_c = \frac{\sum_j^k (c_j)}{k}$$

If $k > T_6$ then a candidate is allowed to be either be dropped or promoted. If $A_c > T_3$ the candidate is promoted to a match. Meanwhile, if $A_c < T_4$ the candidate may be dropped from the pool as an anomaly.

Secondly, it is possible that a candidate is erroneously promoted. To address this, we can demote a phrase to candidate status if enough users are uncertain of it. We again consider the average of the certainty scores A_m but this time of the k passing users who considered the phrase M in some problem (M, C, R). Similarly, if m_j is the certainty score of user j normalized to a value between -1 and 1 and the average is

$$A_m = \frac{\sum_j^k (m_j)}{k}$$

then a match can be demoted if $A_m < T_7$ (and $k > T_6$).

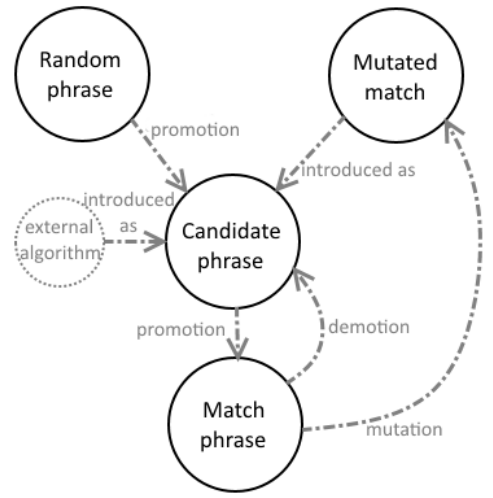


Figure 9: The states a phrase can go through in the system.

4.8 Finding synonyms

We can identify likely synonyms of words when match phrases are mutated (creating candidate phrases). When a word is replaced in the match phrase, the substitute word can be considered a likely synonym if the new mutated phrase is promoted to a match.

To achieve this, we track which word was substituted in a given candidate. If the candidate is promoted to a match, we add the pair $(substitute, word)$ to a list of synonyms found.

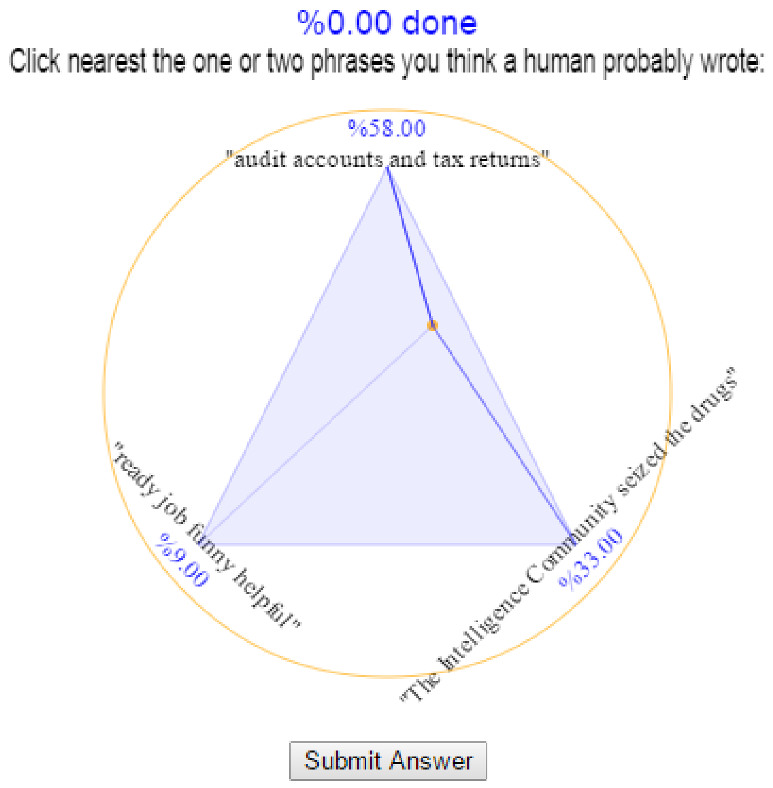

%0.00 done
Click nearest the one or two phrases you think a human probably wrote:
%58.00
"audit accounts and tax returns"
"ready job funny helpful"
%9.00
"The Intelligence Community seized the drugs"
%33.00
Submit Answer

Figure 10: An example testing session.

5 System evaluation

5.1 Criteria

Overall, the quality of the system should be evaluated on the merits of its usability and security. Secondarily, it can be evaluated on the quality of its semantic data contributions.

5.2 Security analysis

The scheme may achieve subpar results if a hypothetical attacker has access to the same initial database. Fritsch et al. [2010] discusses the flaws of a CAPTCHAs which relies on a hidden database or answer key, rather than generating problems on the fly. This system is something of a hybrid approach, using a database to generate new problems. It is worth noting that the issues of maintaining secrecy of a database is mitigated by the fact that the system's database is not static. Any given instance of the running system would have a distinct database that would be instantaneously updated with every accepted user. To this end, though, it is important to cultivate a diverse set of candidate phrases, as discussed above.

To evaluate the system, trials with human subjects will be compared against a brute force attack. The major metrics to be used will include acceptance rate of human users and rejection rates of brute force attacks. This is a straightforward and typical way to evaluate CAPTCHA systems. If these acceptance and rejection rates are comparable to similar CAPTCHAs, this will provide a measure of the viability and security of the system respectively.

For an initial test, we performed more than 20 user sessions (culminating in rejection or acceptance). A minimum bar for success was that the system rejected no more than 10 percent of the time with human users. While initial human tests were adequate, showing no rejections, the average number of attempts was initially between 2 and 3.

We compared these results with 1000 sessions with a brute force (choos-

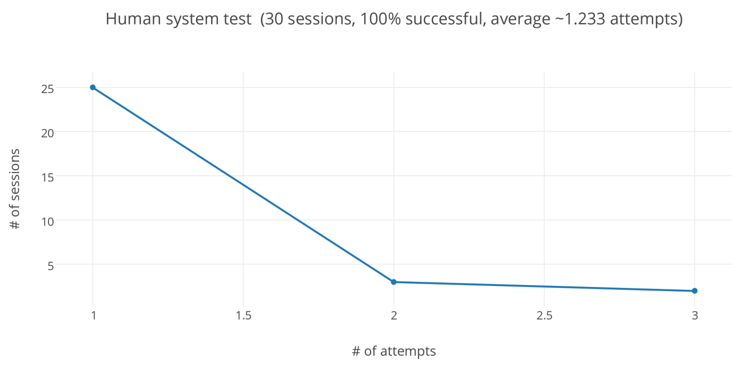


Figure 11: Distribution of attempts over all sessions of preliminary testing with humans.

ing solution randomly) attack. Results from these tests together implied that human responses were typically very close to the correct phrase (or mutated phrase), while random responses of course had a uniform distribution. Because of this observation, our original degree of quality calculation (which was linear) was substituted for a new nonlinear one (highly correct and highly incorrect responses are weighted more heavily).

We ran the same tests again after implementing the new quality calculation. Figure 11 and Figure 12 show these results. The average human user session with the revised algorithm finished with approximately 1.233 attempts. Sessions with the brute force attacker had a similar statistic, but with a drastically lower rate of success (%0.9 versus %100). This indicates a minimum level of security which we find adequate for a prototype.

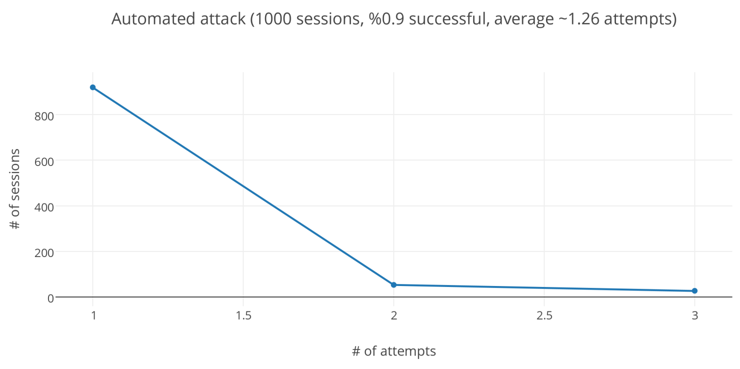


Figure 12: Distribution of attempts over all sessions of preliminary testing versus a brute force attacker.

5.3 Usability analysis

The system must be evaluated on grounds of usability, particularly whether the prompt can be easily understood by users. Of course, the user's grasp of the graphical layout and meaning of their choice is as essential to authentication as feedback.

For usability evaluation, the most important numerical measures besides failure rate are the average number of attempts required for a human user to be accepted, and the overall time spent for the user to complete the problems given (in seconds). These are among the criteria that Bursztein et al. [2010] uses to evaluate a large number of CAPTCHAs in regular use. They observe that systems vary widely in terms of usability (”easiness” for humans), with audio-based CAPTCHAs being the most ”difficult”. In our limited testing of this system, the human solving time was consistently between 10-20 seconds. This is consistent with the solving time Bursztein et al. [2010] observed for popular image-based CAPTCHAs (which one can assume are fairly familiar tasks to solve).

For this system, intuitively the time to solve a given problem would seem to be proportional to the length of phrases chosen (time to read). For this reason, shorter phrases may be slightly preferred.

We must also determine if stochastically generated short phrases will too often have meaning for users, causing confusion by generating problems with only meaningful phrases as choices. This could be mitigated by choosing longer phrases, which would decrease the probability that a given set of randomly selected words will have a coherent meaning. In a preliminary test, the author has found that in a sample of 100 generated problem cases with phrases of 3 words or more, every randomly generated phrase appears subjectively less meaningful than the match phrase it is coupled with and only 3 such phrases even have the appearance of a sentence fragment (see Appendix C for a listing of the data). This provisionally indicates that lengthier phrases are not needed.

6 Development details

6.1 Implementation and technologies used

To provide the GUI for the client, a custom JavaScript/HTML5 canvas interface is used. The interface responds to any mouseclick within the designated triangular area. Any such choice of position constitutes the user's solution

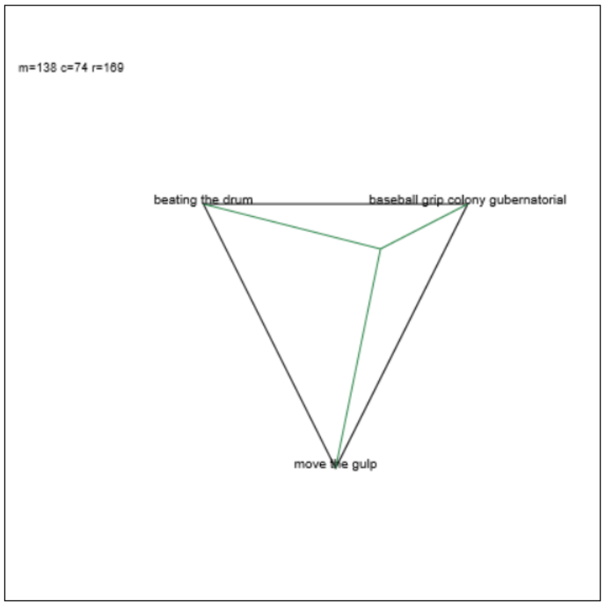


Figure 13: An early alpha of the GUI in debug view, showing distances between vertices.

to the presented problem. A separate submission button will be included to allow the user to fine-tune their answer before sending it to the server.

On the server side, Node.js is used to handle incoming connections and generate problem cases for users. For WordNet access, the *WNdb* (WordNet database) Node.js package is employed. This package provides direct retrieval of synsets from the database by lemma (synset element).

Node.js is naturally non-blocking, asynchronous and event-driven, with callbacks typically used to finish tasks. Node.js is appropriate because it is designed to handle many concurrent client connections over HTTP, making the system in principle scalable beyond the size we test the system at. See Figure 15 for a diagram of how the server interacts with the clients and its stored data.

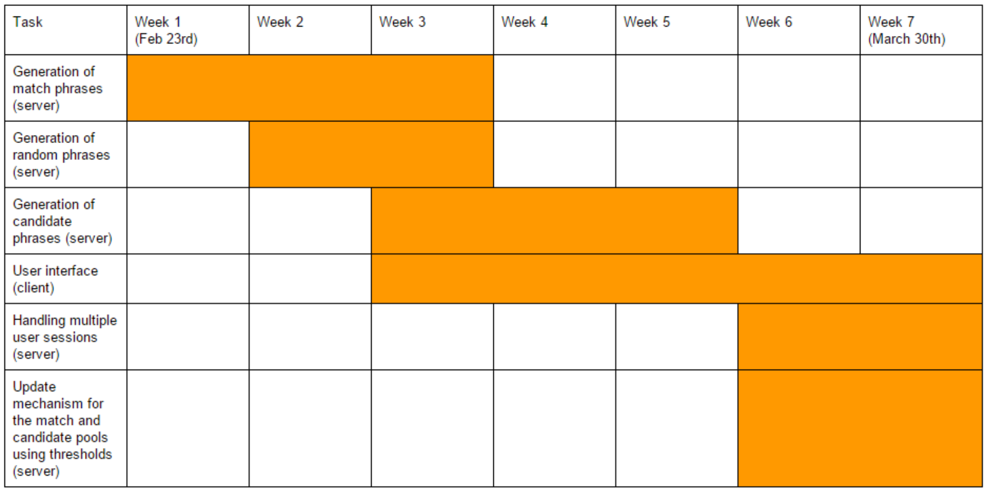
During a user session, the server may produce one or more problems, consisting of a generated match, candidate, and random phrase. If a user answers the problem(s) sufficiently correctly, the server will accept the user and the session ends. Alternately, the session may end if the user is rejected. The update mechanism, described in more detail in the methodology section above, is triggered only upon the acceptance of a user session.

6.2 Major system components and timeline

The system involves a server and client segment. The major tasks/components of the development are:

- Generation of match phrases (server)
- Generation of random phrases (server)
- Generation of candidate phrases (server)
- Handling multiple user sessions (server)
- Update mechanism for the match and candidate pools using thresholds (server)
- User interface (client)

A timeline for the implementation of these components was created as seen in Figure 14.



Task	Week 1 (Feb 23rd)	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7 (March 30th)
Generation of match phrases (server)	🟧	🟧	🟧				
Generation of random phrases (server)		🟧	🟧				
Generation of candidate phrases (server)			🟧	🟧	🟧		
User interface (client)			🟧	🟧	🟧	🟧	🟧
Handling multiple user sessions (server)						🟧	🟧
Update mechanism for the match and candidate pools using thresholds (server)						🟧	🟧

Figure 14: A development timeline (Gantt) chart.

6.3 Description of development approach

For the software development, an Agile coding methodology is employed. An Agile coding methodology is characterized by an iterative and evolutionary style. Each iteration is completed in a relatively short time frame, involving stages of design, coding and acceptance testing. In this project, after the completion of a major component of the software it is immediately subjected to so-called smoke testing to ensure the component functions at a basic level. This is followed by an integration test with the other components completed so far.

To clarify, there is also an Agile project management approach, which emphasizes responsiveness to clients. The client provides ”user stories” describing the functionality they would like to be implemented initially. However, these guides for development which allow more flexibility in the end

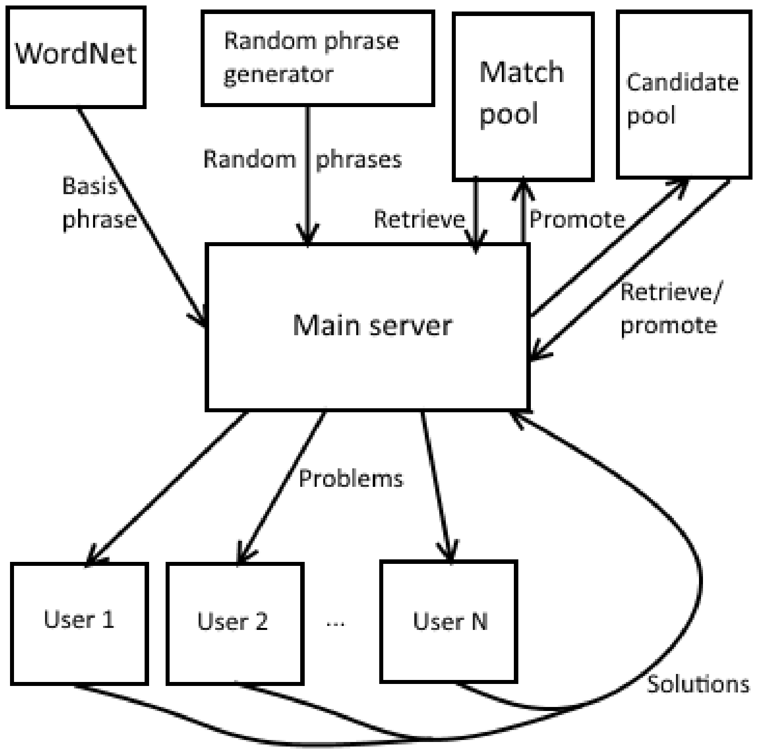
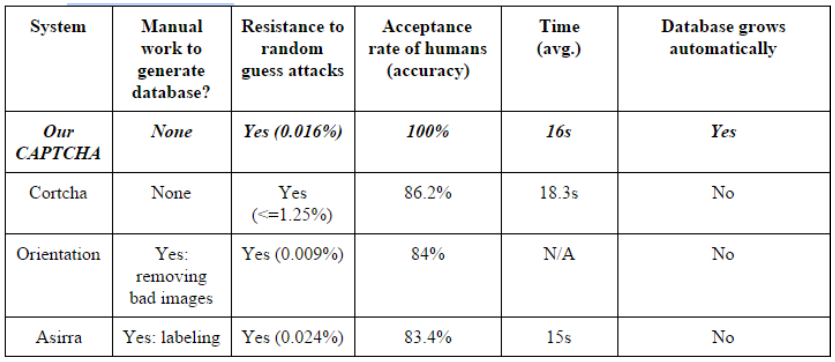


Figure 15: Interaction between server, clients, and stored data.

product than in a traditional waterfall approach. The short length of iterations allows regular feedback, and direct input from the client on the progress of development. In the case of a self-directed research project such as this one, the feedback stage takes the form of an informal analysis of intermediate results by the author and the supervising professor.



System	**Manual work to generate database?**	**Resistance to random guess attacks**	**Acceptance rate of humans (accuracy)**	**Time (avg.)**	**Database grows automatically**
Our CAPTCHA	***None***	***Yes (0.016%)***	***100%***	***16s***	***Yes***
Cortcha	None	Yes (<=1.25%)	86.2%	18.3s	No
Orientation	Yes: removing bad images	Yes (0.009%)	84%	N/A	No
Asirra	Yes: labeling	Yes (0.024%)	83.4%	15s	No

Figure 16: Results in context of comparable recent CAPTCHA designs from the image recognition family.

7 Conclusions

7.1 Comparing results

Language-based CAPTCHAs have not been well-explored, so we compare our system to examples from the image recognition family. Many examples from this family (and language-based systems like SemCAPTCHA) rely upon manual human work to create a working database from which to generate challenges for users. To increase the size of our database, one must merely run the system over a longer period of time.

The rule of thumb for an acceptable level of resistance to blind guess attacks varies, but Zhu et. al suggest a level of 0.6%,which we easily accomodate. In terms of resistance to random guess attacks and in acceptance rate for humans our system is competitive or superior with comparable systems.

The system could be adapted to identify semantic relationship between words, as WordNet's data does. Extensions of the concept using a similar multilevel, triangular probability scheme could work to identify the relational type of a phrase. This would be at a tradeoff of additional prompts for users, which would ideally be generally avoided.

Combination of the proposed method with a traditional optical text recognition approach employing visual distortion on the displayed phrases may reduce accessibility for blind persons, for example, but present a more thorough defense.

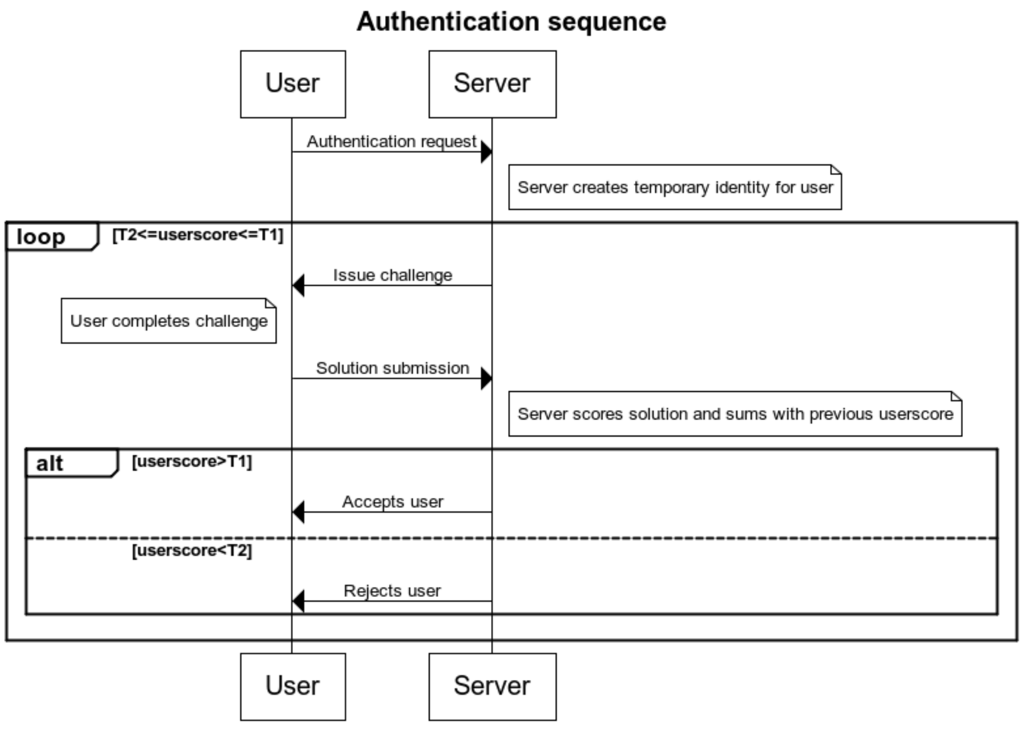
One other significant security concern is increasing the difficulty of a dictionary-style attack on the system. The sensitivity of the responses to repeated wrong answers minimizes the success of completely blind attacks, as shown in our early tests, but some valid worry remains regarding more sophisticated approaches. For example, an attack based on search engine query results may be more effective.

More study may also be required to find an ideal configuration of threshold values based on the various requirements of phrase pool diversity, user reaction to the sensitivity level, and the quality and rate of accumulating match phrases or synonyms. However, good estimation of these thresholds might require a deployment of the software package to some sizeable userbase.

7.2 Implications and summary

We presented some philosophy behind CAPTCHA design that would support qualitative rather than absolute criteria for evaluation, and designed a system with those goals in mind. The probabilistic nature of the system provides a desired granularity of responses that many CAPTCHAs do not and the collection of responses (feedback) allows for the generation of new semantic data. It is indicated that a probabilistic, language-based CAPTCHA system is a feasible means to secure a web service.

8 Appendix A: Main authentication sequence UML



9 Appendix B: Sample code

The following is a partial listing of some of the most relevant portions of the software code. The full code can be found on GitHub at this address:

`https://github.com/teslinroys/wc_captcha`

It is published under the MIT License, see LICENSE file for details. For code documentation, TypeDoc is used for the TypeScript code and Docco for the Node.js code.

9.1 Serverside (Node.js)

```
function genProblem(callback)
{
    var triplet = new Array();
    var phrase_functions = new Array();
    phrase_functions.push(genMatchPhrase);
    phrase_functions.push(genCandidatePhrase);
    phrase_functions.push(genRandomPhrase);

    phrase_functions.forEach(function (phrasegen_func) {
        phrasegen_func(getRandomInt(3,7), function (phrase) {
            triplet.push(phrase)
            if (triplet.length == 3)
                callback(triplet[0]+", "+triplet[1]+", "+triplet
                    [2]);
        });
    });
}

function genRandomPhrase(plength, callback)
{
    var rw = rndwords(plength);
    var phrase = "\"" + rw[0];
    for (var i = 1; i < plength; i++)
        phrase = phrase + " " + rw[i];
    phrase = phrase + "\"";
    callback(phrase);
}

function genMatchPhrase(plength, callback)
{
    var phrase_options = new Array();
    var lemma = verblist[getRandomInt(0, verblist.length)]; //
        Keep generating random verb lemmas
    wordnet.lookup(lemma, function (results) { //And doing
        WordNet lookups for the synsets the lemma is a part of
        results.forEach(function (result) { //For each synset we
            find,
                var gl = String(result.gloss); //Extract
                    glossary from synset
                var sentence_pattern = /"(.*?)"/g; //Use a
                    regular expression to identify the sentence
                    examples
                var matches = gl.match(sentence_pattern); //Find
                    matches for the regex
                if (matches != null) //If there are any matches,
                    for (var n = 0; n < matches.length; n++) //
                        Go through each match
                        if (tokenizer.tokenize(matches[n]).
                            length == plength) //And if it is the
                            right length
                            phrase_options.push(matches[n]); //
                                Make a note of the phrase
            //}
        })
        if (phrase_options.length < 1)
            genMatchPhrase(plength, callback); //try another
                lemma if this one didn't give us the results we
                want
        else
            callback(phrase_options[getRandomInt(0,
                phrase_options.length)]); //complete when at
                least one phrase to choose from is found
    })
}

function genCandidatePhrase(plength, callback) {
    genMatchPhrase(plength, function mutate(phrase) {
        var phrasewords = tokenizer.tokenize(phrase);
        var mutations = new Array();
        var countwords = 0;
        phrasewords.forEach(function wordInPhrase(word) {
            mutateWord(word, function givenMutation(word,
                mutation) {
                if (word!=mutation)
                    mutations.push([word, mutation]);
                countwords++;
                if (countwords == phrasewords.length)
                    callback(applyMutation(phrasewords,
                        mutations[getRandomInt(0, mutations.
                        length)]))
            })
        });
    });
}

function applyMutation(phrasewords, mutation)
{
    var mp = "\"" + phrasewords[0];
    for (var i = 1; i < phrasewords.length; i++)
        mp = mp + " " + phrasewords[i];
    mp = mp + "\"";
    if (mutation!=null)
        mp=mp.replace(mutation[0], "("+mutation[0]+"->"+mutation
            [1]+")");
    return mp;
}
```

```

function mutateWord(word, callback)
{
    wordnet.lookupSynonyms(word, function (synonyms) {
        if (synonyms.length > 0) {
            var pos_matching_syns = new Array();
            var countsyns = 0;
            synonyms.forEach(function synInSet(syn) {
                samePOS(word, syn.lemma, function (isSame) {
                    countsyns++;
                    if (isSame)
                        pos_matching_syns.push(syn);
                    if (countsyns == synonyms.length) {
                        if (pos_matching_syns.length > 0)
                            callback(word, pos_matching_syns[
                                getRandomInt(0, pos_matching_syns
                                .length)].lemma);
                        else
                            callback(word, word);
                    }
                })
            });
        }
        else
            callback(word, word);
    });
}
```

9.2 Clientside (Typescript/HTML5)

```
class Vector2 {
    public X: number;
    public Y: number;
     /**
 * Represents a 2D vector.
 */
    constructor(x: number, y: number) {
        this.X = x;
        this.Y = y;
    }
    /** Returns the distance between this vector and the input
        vector. */
    distanceTo(v2: Vector2): number {
        var a = this.X - v2.X;
        var b = this.Y - v2.Y;
        var c2 = Math.pow(a, 2) + Math.pow(b, 2);
        return Math.sqrt(c2);
    }
}

class Captcha {
    public con: CanvasRenderingContext2D;
    public canvas: HTMLCanvasElement;
    public control_pts:Vector2[];
 /**
 * Constructs the interactive CAPTCHA element.
```

```
*/
    constructor(c: HTMLCanvasElement) {
        this.con = c.getContext('2d');
        this.canvas = c;
        this.canvas.addEventListener("mousedown", (event:
            MouseEvent) => this.onMouseDown(event), false);
        this.control_pts = [new Vector2(250, 50), new Vector2
            (50, 450), new Vector2(450, 450)];
    }
    /** This event handler redraws the canvas when it is clicked
        . */
    onMouseDown(e: MouseEvent) {
        var x, y;
        if (e.pageX || e.pageY) {
            x = e.pageX;
            y = e.pageY;
        }
        else {
            x = e.clientX + document.body.scrollLeft +
            document.documentElement.scrollLeft;
            y = e.clientY + document.body.scrollTop +
            document.documentElement.scrollTop;
        }
        // Convert to coordinates relative to the convas
        x -= this.con.canvas.offsetLeft;
        y -= this.con.canvas.offsetTop;

        var mp: Vector2 = new Vector2(x, y);
        var m: Vector2 = this.control_pts[0];
        var c: Vector2 = this.control_pts[1];
        var r: Vector2 = this.control_pts[2];
        var b: boolean = this.pointInTriangle(mp, m, c, r);
        this.draw();
        if (b == true) {
            this.con.fillText("m=" + Math.round(mp.distanceTo(m)
                ) +" c=" + Math.round(mp.distanceTo(c)) + " r=" +
                 Math.round(mp.distanceTo(r)), 50, 50, 80);
            this.con.beginPath();
            this.con.moveTo(m.X, m.Y);
            this.con.lineTo(mp.X, mp.Y);
            this.con.moveTo(c.X, c.Y);
            this.con.lineTo(mp.X, mp.Y);
            this.con.moveTo(r.X, r.Y);
            this.con.lineTo(mp.X, mp.Y);
            this.con.closePath();
            this.con.strokeStyle = 'rgb(32, 128, 64)';
            this.con.stroke();

        }
    }
    /** Draws the CAPTCHA element to the specified canvas. */
    draw() {
        this.con.clearRect(0, 0, this.con.canvas.width, this.con
            .canvas.height);
        this.con.beginPath();
        this.con.moveTo(this.control_pts[0].X, this.control_pts
            [0].Y);
        for (var i = 0; i < this.control_pts.length; i++) {
            this.con.lineTo(this.control_pts[i].X, this.
                control_pts[i].Y);
        }
        this.con.lineTo(this.control_pts[0].X, this.control_pts
            [0].Y);
        this.con.closePath();
        this.con.strokeStyle = 'black';
        this.con.stroke();
    }
    /** Returns the cursor position relative to the canvas. */
    getCursorPosition(e) {
    var x;
    var y;
    if (e.pageX || e.pageY) {
        x = e.pageX;
        y = e.pageY;
    }
    else {
        x = e.clientX + document.body.scrollLeft +
        document.documentElement.scrollLeft;
        y = e.clientY + document.body.scrollTop +
        document.documentElement.scrollTop;
    }
    // Convert to coordinates relative to the convas
    x -= this.con.canvas.offsetLeft;
    y -= this.con.canvas.offsetTop;

    return [x, y]
}

    /** Returns whether or not a given point (e.g. mouse
        position) is inside a triangular area. */
    pointInTriangle (point:Vector2, v1:Vector2, v2:Vector2, v3:
        Vector2) : boolean {
        var A = (-v2.Y * v3.X + v1.Y * (-v2.X + v3.X) + v1.X * (
            v2.Y - v3.Y) + v2.X * v3.Y) / 2;
        var sign = A < 0 ? -1 : 1;
        var s = (v1.Y * v3.X - v1.X * v3.Y + (v3.Y - v1.Y) *
            point.X + (v1.X - v3.X) * point.Y) * sign;
        var t = (v1.X * v2.Y - v1.Y * v2.X + (v1.Y - v2.Y) *
            point.X + (v2.X - v1.X) * point.Y) * sign;
        return s > 0 && t > 0 && s + t < 2 * A * sign;
    }
}

var captcha;

/** Loads CAPTCHA element. */
window.onload = () => {
    var c = <HTMLCanvasElement> document.getElementById('captcha
        ');
    captcha = new Captcha(c);
    captcha.draw();
};
```

10 Appendix C: Test data

In the following data, only entries 29, 44, and 81 contain a random phrase with even fragmentary meaning. Notation (a-¿b) represents a mutation in a candidate phrase (would not be visible in a deployed system).

```
0 "concerned captain clear material particularly battle", "the (
    sommelier->waiter) decanted the wines", "mangle the sheets"
1 "face table given crowd", "The event evades explanation", "His
    (eyes->opinion) were glistening"
2 "foreign comfortable upper", "some salts sublime when heated",
    "We breakfast at (seven->digit)"
3 "grandmother shop pay outside people tax", "Count on the
    monsoon", "Mendel (tried->reliable) crossbreeding"
4 "trip language season occasionally sea friend", "was
    discontented with his position", "she achieved her (goal->
    own_goal) despite setbacks"
5 "rule between difficulty mood manner feet", "relieve the
    pressure and the stress", "The Turks besieged (Vienna->
    schnitzel)"
6 "lungs worry merely act bear anything", "install the washer
```

```
    and dryer", "she could not (forbear->refrain) weeping"
7 "girl want sale cotton shirt unit", "Jupiter has sixteen moons
    ", "a league of (disunited->divided) nations"
8 "straight brief sign mass pie must", "Sample the regional
    dishes", "(Murdoch->writer) owns many newspapers"
9 "sum bright listen fruit paragraph highest", "they performed
    with great polish", "The sun shone on the (fields->comedian)"
10 "paid steam suppose mountain trick taste", "We embarked on an
    exciting enterprise", "(repel->disgust) the enemy"
11 "fairly cell greater", "The children crunched the (celery->
    celery_stick) sticks", "a poster advertised the coming
    attractions"
12 "changing beginning birthday smallest only finest", "hose the
    lawn", "a tincture of (condescension->arrogance)"
13 "hearing pain mirror", "Her face radiated with happiness", "
    She leaned over the (banister->baluster)"
14 "send you amount won", "batten down a ship's hatches", "zap
    the (enemy->people)"
15 "saved tone felt voyage general", "pray to the (Lord->
    margrave)", "the final draft of the constitution"
16 "bar flow against", "The ceiling blackened", "the oven is (
    off->execute)"
17 "room customs full", "intertwine the ribbons", "She
    understands (French->Walloon)"
18 "exactly above minute tropical", "nothing will result from
    this meeting", "an uncut (diamond->minor_suit)"
19 "command vast fix shape", "grout the (bathtub->footbath)", "
```

```
    We had to see a psychiatrist"
20 "hunt fly bottle mathematics", "her health is better now", "
    This car suits my (purpose->aim) well"
21 "park flow stronger", "glass the windows", "the (reticulate->
    fretted) veins of a leaf"
22 "pressure meal its previous lack", "She called for room
    service", "The musical (performance->rendition) sparkled"
23 "page pure ocean", "hyphenate these words and names", "The
    horse (finally->last) struck a pace"
24 "bow metal sick moon later", "The curtain swooshed open", "
    The two old friends (paired->mated) off"
25 "friendly improve soon already minerals", "The circumstances
    extenuate the crime", "a deafening (noise->bark)"
26 "regular fellow perfect laid trouble", "This question really
    stuck me", "you can rely on his (discretion->prudence)"
27 "fact change stepped function farmer fall", "a sizable
    fortune", "(Don->United_Kingdom) t belittle his influence"
28 "that pound having immediately", "a (schoolgirl->female_child
    ) fantasy", "Her dream really materialized"
29 "scared about time yourself", "splat fish over an open fire",
     "Sorry to (trouble->scandal) you but"
30 "vegetable congress fifteen observe prevent", "Which (horse->
    stepper) are you backing", "Storm the fort"
31 "food other size spell", "nominate a (committee->
    fairness_commission)", "The oil suppurates the pustules"
32 "vessels eye face trunk like pleasure", "The man scrupled to
    perjure himself", "(slosh->splash) paint all over the walls"
```

```
33 "certain jar sink", "She pursued many activities", "He refused my (offer->overbid) of hospitality"
34 "street alike visit hurried donkey", "the dog chased the rabbit", "(Some->few) cells had nucleated"
35 "watch is variety split", "bring out the truth", "These herbs (suffer->grieve) when sunned"
36 "customs search morning planning addition danger", "the board was a foot short", "(He->noble_gas) managed to multiply his profits"
37 "tide throughout lying", "The colors blend well", "The boiling soup was (frothing->unhealthy)"
38 "eight tears trip", "He hit a home run", "the (dropout->individualist) rate"
39 "chart fun wore solve everything", "skilled in the utilization of computers", "Her savings dwindled (down->lowered)"
40 "pitch pot cut yet problem cat", "the ship beached near the port", "not (worth->pennyworth) shucks"
41 "somehow poet pink changing duty", "he extended his mitt", "pursue a (hobby->speleology)"
42 "potatoes disappear donkey", "ironing gets rid of most wrinkles", "(I->halogen) aim to arrive at noon"
43 "one old north it coach respect", "The chemical undergoes a sudden change", "a belt of (high->elated) pressure"
44 "definition factory image", "The storm fused the electric mains", "This (statement->answer) misrepresents my intentions"
```

```
45 "statement fill music complete vote foreign", "He quoted the
   Bible to her", "The (army->standing_army) surged forward"
46 "family fully room", "The home team scored many times", "that
    mountain is (solid->hard) rock"
47 "fur whole atmosphere concerned grown house", "mill a coin",
   "a (wet->drippy) bathing suit"
48 "effect potatoes instant sister", "it was all for naught", "
   The (vessel->bathtub) hove into sight"
49 "fierce had position did motion made", "(I->chemical_element)
    misplaced my eyeglasses", "erupt in anger"
50 "heat drew gather fox century key", "(Adam->man) knew Eve", "
   core an apple"
51 "push mail building", "a flight destined for New York", "
   retell a (story->fib)"
52 "report hang rays", "sanctify the marriage", "a life (
   consecrated->desecrated) to science"
53 "recently point direction solve bone", "This will save money"
   , "The (boat->barge) tacked"
54 "thirty per variety greatest fell swing", "The dining room
   blackened out", "(Ring->association) the bells"
55 "proud slabs leave count", "Snow capped the mountains", "The
   (bad->severe) witch cursed the child"
56 "studying whole only", "The branches made a roof", "what he
   said was mostly (bull->fake)"
57 "using damage camera range", "she was sporting a (new->
   original) hat", "We had to oblige him"
58 "bush nine pet cup", "epoxy the shards", "(I->seawater)
```

```
    apologized for being late"
59 "public clean coal rear bottom", "This experience transformed
     her completely", "sugar your (tea->oolong)"
60 "writing shaking straight hidden", "sleep off a hangover", "
    the party went with a (bang->lover)"
61 "door bean wheel at heading", "His (sharp->penetrative) nose
    jutted out", "check the brakes"
62 "happily mix busy principle rubber", "Interested tapered off"
    , "(I->letter) managed his campaign for governor"
63 "save season means common floating be", "dancers in two
    parallel rows", "These signs bode bad (news->soft_news)"
64 "teeth lips alphabet", "The pollution is endangering the
    crops", "Stay with me (please->delight)"
65 "spread song wealth universe discovery", "he always follows
    the (latest->fashionable) fads", "this may land you in jail"
66 "few whispered movement", "a yet sadder tale", "(bandy->unfit
    ) about an idea"
67 "rich slightly crowd alone into", "marbleize the fireplace",
    "a twofold (increase->addition)"
68 "musical twelve whose well common", "wanton one s (money->
    appropriation) away", "disarrange the papers"
69 "accident these roll force safety", "She called for room
    service", "the (novel->novelist) had chapter titles"
70 "ten indeed met hurt problem moment", "she gave a gasp and
    fainted", "(sectionalize->divide) a country"
71 "sang earn likely softly afternoon principle", "The road
    divaricates here", "raft wood down a (river->Niagara)"
```

```
72 "breathe trap hundred", "I want my own room", "The (war->Arab
   -Israeli_War) desensitized many soldiers"
73 "voice freedom frighten", "Please bracket this remark", "we
   had a good (discussion->argument)"
74 "night ever bring drink say addition", "approach a new
   project", "The (mud->soil) mired our cart"
75 "structure known border whole", "Bullets spanged into the
   trees", "The villa dominates the (town->Gadsden)"
76 "court particularly born raise", "The parents had the child
   baptized", "The bread (crusted->covered) in the oven"
77 "difficult store halfway underline", "We were hiking in
   Colorado", "You cannot (believe->understand) this man"
78 "diameter blue frame save together memory", "The life vest
   buoyed him up", "File these bills (please->satisfy)"
79 "number better welcome measure electric second", "I blundered
    during the job interview", "the (compound->enamel)
   decarboxylated"
80 "brown famous flies", "The spaceship blazed out into space",
   "Tone down that aggressive (letter->psi)"
81 "combination climate of city", "He gripped the steering wheel
   ", "leather (bound->city_line) volumes"
82 "studied get sky short trunk", "militarize the Civil Service"
   , "Walk the (tightrope->rope)"
83 "rod prevent gasoline star situation", "Her gesture
   emphasized her (words->line)", "The students filed into the
   classroom"
84 "occur dollar setting exciting setting", "(He->
    Hebrew_alphabet) tried to crystallize his thoughts", "She
    unsexed herself"
85 "selection salmon met bit body", "preserve the (peace->order)
     in the family", "chrome bathroom fixtures"
86 "freedom vessels court", "This drug expectorates quickly", "(
    bend->curl_up) the rod"
87 "plus scene selection shine", "the peculiar aromatic odor of
    cloves", "The pornographic pictures sickened (us->Colony)"
88 "plant gift salt captain easy twice", "You must overcome all
    difficulties", "This medicine is (home->homeowner) confected"
89 "rise sleep place", "They buried the stolen goods", "(I->
    seawater) m beat"
90 "possible hour dream cross wait ring", "a building of vast
    proportions", "(admit->reject) someone to the profession"
91 "after useful surrounded sat mixture limited", "ruffle
    somebody's composure", "the farmers overcropped the (land->
    Saint_Kitts_and_Nevis)"
92 "relationship brain represent discover", "The student rewrote
     his thesis", "trees line the (riverbank->bank)"
93 "pattern stomach thread themselves discussion immediately", "
    the Welsh coast", "please mark the (individual->causal_agent)
     pages"
94 "young single wise", "(pit->opponent) plums and cherries", "
    the engine is idling"
95 "rod evening welcome", "This statement misrepresents my
    intentions", "(don->scarf) t get in a stew"
96 "wave than pack hang travel hunt", "A hot soup will revive me
```

```
    ", "Where do these books (go->act)"
97 "all pond window degree", "Charlie likes to play Mary", "let
    s give it a (whirl->spinner)"
98 "consider brought fifth tube equipment win", "spin a coin", "
    the (court->bench) set aside the conviction"
99 "unknown part strength broke", "avert a strike", "(play->
    action) a joke"
```

References

Abdulaziz S Almazyad, Yasir Ahmad, and Shouket Ahmad Kouchay. Multimodal captcha: A user verification scheme. In *Information Science and Applications (ICISA), 2011 International Conference on*, pages 1–7. IEEE, 2011.

Henry S Baird and Terry P Riopka. Scattertype: a reading captcha resistant to segmentation attack. In *Electronic Imaging 2005*, pages 197–207. International Society for Optics and Photonics, 2005.

Elie Bursztein, Steven Bethard, Celine Fabry, John C Mitchell, and Daniel Jurafsky. How good are humans at solving captchas? a large scale evaluation. In *Security and Privacy (SP), 2010 IEEE Symposium on*, pages 399–413. IEEE, 2010.

Elie Bursztein, Matthieu Martin, and John Mitchell. Text-based captcha strengths and weaknesses. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 125–138. ACM, 2011.

Monica Chew and J Doug Tygar. Image recognition captchas. *Information Security*, pages 268–279, 2004.

Christoph Fritsch, Michael Netter, Andreas Reisser, and Günther Pernul. Attacking image recognition captchas. In *Trust, Privacy and Security in Digital Business*, pages 13–25. Springer, 2010.

Kurt Gödel. Über formal unentscheidbare sätze der principia mathematica und verwandter systeme i. *Monatshefte für mathematik und physik*, 38(1): 173–198, 1931.

Rich Gossweiler, Maryam Kamvar, and Shumeet Baluja. What's up captcha?: a captcha based on image orientation. In *Proceedings of the 18th international conference on World wide web*, pages 841–850. ACM, 2009.

Carlos Javier Hernandez-Castro, Arturo Ribagorda, and Julio Cesar Hernandez-Castro. On the strength of egglue.

David Hilbert and Wilhelm Ackermann. Grundzüge der theoretischen logik. *Berlin, Heidelberg*, 1928.

Graeme Hirst and Alexander Budanitsky. Correcting real-word spelling errors by restoring lexical cohesion. *Natural Language Engineering*, 11(01):87–111, 2005.

Paweł Łupkowski and Mariusz Urbanski. Semcaptcha—user-friendly alternative for ocr-based captcha systems. *Speech and Language Technology*, 11:278–289, 2008.

John McCarthy. Review of the emperor's new mind by roger penrose. *Bulletin of the American Mathematical Society*, 23(2):606–616, 1990.

M Mehra, M Agarwal, R Pawar, and D Shah. Mitigating denial of service attack using captcha mechanism. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pages 284–287. ACM, 2011.

George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

Greg Mori and Jitendra Malik. Recognizing objects in adversarial clutter: Breaking a visual captcha. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–134. IEEE, 2003.

Gabriel Moy, Nathan Jones, Curt Harkless, and Randall Potter. Distortion estimation techniques in solving visual captchas. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–23. IEEE, 2004.

Roger Penrose. *The emperor's new mind: concerning computers, minds, and the laws of physics*. Oxford University Press, 1999.

Hilary Putnam. Models and reality. *The Journal of Symbolic Logic*, 45(03): 464–482, 1980.

Narges Roshanbin and James Miller. A survey and analysis of current captcha approaches. *Journal of Web Engineering*, 12(1-2):1–40, 2013.

Steven A Ross, J Alex Halderman, and Adam Finkelstein. Sketcha: a captcha based on line drawings of 3d models. In *Proceedings of the 19th international conference on World wide web*, pages 821–830. ACM, 2010.

Amalia Rusu and Venu Govindaraju. Handwritten captcha: Using the difference in the abilities of humans and machines in reading handwritten words. In *Frontiers in Handwriting Recognition, 2004. IWFHR-9 2004. Ninth International Workshop on*, pages 226–231. IEEE, 2004.

John R Searle. Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424, 1980.

Alan M Turing. Computing machinery and intelligence. *Mind*, pages 433–460, 1950.

Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58:345–363, 1936.

Steven Vercruysse. Wordvis: the visual dictionary. `http://wordvis.com/`, 2010.

Luis Von Ahn, Manuel Blum, Nicholas J Hopper, and John Langford. Captcha: Using hard ai problems for security. *Advances in Cryptology—EUROCRYPT 2003*, pages 294–311, 2003.

Luis Von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.

Amber Wilcox-O'Hearn, Graeme Hirst, and Alexander Budanitsky. Real-word spelling correction with trigrams: A reconsideration of the mays, damerau, and mercer model. *Computational Linguistics and Intelligent Text Processing*, pages 605–616, 2008.

Rong Zhao and William I Grosky. Negotiating the semantic gap: from feature maps to semantic landscapes. *Pattern Recognition*, 35(3):593–600, 2002.