My Word! Machine versus Human Computation Methods for Identifying and Resolving Acronyms

Christopher G. Harris[1] and Padmini Srinivasan[2]

[1] Dept. of Computer Science, University of Northern Colorado, Greeley CO 80639 USA
[2] Computer Science Department, University of Iowa, Iowa City, IA 52242 USA
christopher.harris@unco.edu

Abstract. Acronyms are commonly used in human language as alternative forms of concepts to increase recognition, to reduce duplicate references to the same concept, and to stress important concepts. There are no standard rules for acronym creation; therefore, both machine-based acronym identification and acronym resolution are highly prone to error. This might be resolved by a human computation approach, which can take advantage of knowledge external to the document collection. Using three text collections with different properties, we compare a machine-based algorithm with a crowdsourcing approach to identify acronyms. We then perform acronym resolution using these two approaches, plus a game-based approach. The crowd and game-based methods outperform the machine algorithm, even when external information is not used. Also, crowd and game formats offered similar performance with a difference in cost.

Keywords: human computation, crowdsourcing, acronym identification, acronym resolution, gamification

1 Introduction

Acronyms are used in many document collections to abbreviate and stress important concepts. The identification of acronyms and discovery of their associated definitions are essential aspects to tasks such as natural language processing of texts as well as knowledge-based tasks such as information retrieval, named entity resolution, ontology mapping, and question-answering. Many people think acronyms are standard (e.g. take the first letter of each word and put them together in all capital letters) but there are many variants from this (e.g., SMART, a German car manufacturer, is an acronym for Swatch + Mercedes + ART; Canola, a type of cooking oil, is an acronym for CANada Oil, Low Acid).

The extraction and resolution of acronyms are not trivial tasks – in many domains, acronyms evolve rapidly. Existing terminological resources and scientific databases cannot keep up to date with the growth of these neologisms. Attempts to manually compose large-scale lexicons of acronym-definition pairs suffer from these same challenges; with such lexicons, information becomes obsolete quickly. Moreover, there are difficulties in the resolution of both ambiguous terms as well as variant forms of the same acronym. For example, Acronym Finder, the world's largest dictionary of acronyms, includes more than 1 million human-edited definitions and is

growing at an average of 5,000 per month [1]; the total effort required to compile this set is estimated to be more than 15,000 hours, a task performed during the last 18 years.

Attending to these shortcomings, this paper evaluates three different non-lexicon approaches to identify and resolve short-form acronyms and their definitions – using machine-based, crowdsourcing-based, and game-based approaches. We evaluate these methods on three text collections with different characteristics: a collection of news articles, a collection of patents, and a collection of journal articles in chemistry. Human-based methods possess advantages that machine methods do not; therefore, our examination focuses on examining where human-based methods provide value, and where they do not, as applied to acronym identification and resolution.

We offer the following contributions. First, we take a string-matching algorithm that has demonstrated good results in one domain and test its effectiveness on three new domains. Second, we examine if a higher number of human assessors provides better accuracy with respect to cost. Last, we examine the improvement offered by two human computation approaches – a game-based approach and a crowdsourcing approach – to see if they are better at finding appropriate definitions in the text and if this comes from knowledge outside of the document.

This paper is organized as follows. In the next section, we provide some background and the motivation behind our work. In Section 3, we introduce our research questions and explain our experimental design. In Section 4, we discuss and analyze our results. Following this, we conclude and provide an assessment of our research questions.

2 Related Research

The goal here of acronym identification and resolution is to extract pairs of short forms (acronyms) and long forms (their expanded forms or definitions) occurring in text. Much of the work with acronyms is either limited to a specific domain (e.g., biomedical text or government documents) or requires the algorithm to be trained on the corpus before use.

In 2003, the TREC Genomics track [2] began a task that invited acronym identification and definition extraction in biomedical text. This TREC-motivated research encouraged the development of a number of algorithms that performed well against biomedical text it was designed to handle. However, few methods used to examine biomedical text have demonstrated their ability to work effectively on text in other domains, a challenge mentioned in [3]. There have been some attempts to use the broader web to extract definitions of terms, such as that by [4] and [5]. Their methods are language independent but are reliant on a large corpus for acronym resolution and may not scale well for documents with rarely-occurring acronyms. In [6], Glass et al. have developed a model that uses Wikipedia to resolve acronyms, but it is dependent on a lexicon of terms. Others, such as [7] have used web mining for detecting acronyms in nursing notes using recurrent neural networking language models (RNNLMs), but this requires a sufficiently large training set in a specialized domain – important because acronyms trained on different domains can lead to dramatically different results.

One challenge in acronym identification and resolution is there are no rules or precise patterns for the creation of acronyms. Moreover, acronyms are ambiguous – the same acronym may refer to two or more different concepts (e.g., *IEM* abbreviates both *immune-electron microscopy* and *interstitial electron model*) and have variant forms (e.g., *NF kappa B*, *NF kB*, NF-KB, *NF-kappaB*, and *NFKB* factor for *nuclear factor-kappa B*). Ambiguity and variation present several challenges in text mining approaches since acronyms have not only must be recognized, but their variants must be linked to the same canonical form and be disambiguated, adding to the complexity of acronym recognition through the use of algorithms.

Schwartz and Hearst [8] implemented an algorithm for identifying acronyms that does not need prior training (unsupervised) using parenthetical expressions as a marker of a short form. In their algorithm, an emphasis is on complicated acronym-definition patterns for cases in which only a few letters match. Once a short-form algorithm is found, they use a fixed-sized window on either side of the term to identify the long form candidate. They make a key assumption: either the long-form or short-form acronym appear in parenthesis in the same sentence. Despite the core algorithm being admittedly simple, the authors report 99% precision and 84% recall on the Medstract gold standard. Because of its simplicity and ease of implementation, this algorithm appears appropriate for generalization to collections outside of biomedicine. Dannélls [9] provided a modified version of the Schwartz and Hearst algorithm, with the advantage of recognizing acronym-definition pairs not indicated by parentheses. They were able to achieve good precision (above 90%) and recall (above 96%) against four Swedish medical text collections. We use Dannélls algorithm as our machine-based approach; pseudocode of the algorithm is provided in Figure 1.

Human-based approaches can add considerable value to acronym identification and resolution. Humans can adapt to the non-standardized rules commonly found in acronym identification and make use of outside knowledge and apply this to acronym resolution. Despite this, there has no work found in the literature that examines crowdsourcing's ability to detect and resolve acronyms, although some other studies have examined named entity resolution (NER) using the crowd, such as that by Finin et al. on Twitter data [10]. In this paper, we explore the value the crowd provides in acronym identification and resolution.

Game formats provide humans with additional incentives to perform tasks well, such as entertainment, challenge, and recognition (i.e., having a successful player's name added to a leaderboard of top scorers). Despite their potential advantages, we have not found any studies exploring the value of games or crowdsourcing as human computation frameworks for acronym identification and resolution.

Motivated by this context, we study the two tasks of (a) Acronym identification, deciding if strings of text are acronyms and (b) Acronym resolution, mapping the short-form acronym onto its long form. For the first task, we compare a machine-based algorithm with a crowdsourcing-based approach. For the second we compare these two approaches and a game-based one. We present results for three text collections that have different properties.

3 Experiment Design

3.1 Research Questions

We investigate the following research questions with respect to acronym identification and resolution:

Q1. Can a machine algorithm developed successfully for one domain also show strong results in other domains? (Note that the algorithm we use does not require training. This is selected intentionally as we wish to stress generalizability to new domains).

Q2. Are improvements in accuracy achieved with human computation methods over a machine-based algorithm? At what financial cost are improvements, if any, achieved?

Q3. Does performance improve when more human participants are involved?

Q4. For acronyms identified correctly by humans, but incorrectly by the algorithm, is the knowledge utilized available in the document, or is it external?

Q5. How can human computation methods help with the difficult-to-resolve acronyms, e.g., those not easy to resolve algorithmically?

3.2 Data

We used 50 documents (for a total of 150), selected randomly from 3 publicly available collections:

- News article documents from TREC collection, disks 4 and 5: LA Times (1989-1990) and Financial Times (1991-1994) newspapers. We used the headline and text/body fields only [11].
- Patent documents come from the MAREC collection, which includes European, Japanese, and US patents, from 2001-2008 [12].
- Chemical Journal articles from four Royal Society of Chemistry journals between 2001 and 2008: Analyst, Journal of Analytical Atomic Spectrometry, Molecular BioSystems, Organic & Biomolecular Chemistry, and Physical Chemistry/Chemical Physics [12].

Characteristic Measured	Chem Journal M	Chem Journal SD	Patent M	Patent SD	News M	News SD
Number of words	988.33	100.39	1078.00	95.95	1601.00	221.57
Number of sentences	42.67	3.75	96.33	15.48	115.00	20.73
Average number of characters per word	5.75	0.09	5.13	0.12	4.79	0.11
Average number of syllables per word	2.01	0.03	1.84	0.04	1.61	0.04
Average number of words per sentence	22.94	0.43	13.19	1.90	14.81	1.09
Gunning-Fog index	18.91	0.33	16.83	0.94	11.22	0.85
Flesch-Kincaid Grade level	17.06	0.29	14.53	0.86	9.23	0.74
ARI (Automated Readability Index)	17.14	0.29	12.66	0.69	8.55	0.95

Table 1. Characteristics of each collection indicating Mean (M) and Standard Deviation (SD)

All documents were in English. Because of their excessive length, if the patent or chemical journal documents exceeded 1000 words, we used the first 1000 words rounded to the end of the paragraph closest to the 1000-word limit. For patents, we used the description field. We did not provide a limit to news articles. Table 1 reports some characteristics of each dataset.

The number of characters and syllables per word from each text collection were not significantly different from each other. However, readability analysis shows that the news articles are the easiest to understand for the general public, whereas the chemistry journals are the most difficult. This was determined consistently both by the Gunning-Fog index (indicating the number of years of formal education a person requires to easily understand the text on the first reading) and the Flesch-Kinkaid and ARI scores. Both are estimates of the U.S. grade level needed to comprehend the text. Readability is important as human computation methods are being tested.

4 Methodology

3.3.1 Gold Standard Data

To create a gold standard acronym list, we had two human assessors who were domain experts each independently evaluate the 150 documents to both identify acronyms and identify their related expansions. Each acronym was counted only once per document, and lists for each assessor for each document were adjudicated over any disagreements. The acronym counts (per document) are provided in Table 2. For example, 185 acronyms were found after adjudication for news of which 183 were resolved. We find more acronyms for the journal collection which is consistent with the earlier observation that this collection is the most difficult (of the 3) to read.

Text Collection	Acronym Identification				Acronym Resolution	
	Assessor 1	Assessor 2	Adjudicated Agreement on Identified Acronyms	Adjudicated Acronyms Requiring Ext. Knowledge to Identify	Adjudicated Agreement on Resolved Definitions	Adj. Acronym Requiring Ext. Knowledge to Resolve
News	189	185	185	11 (5.9%)	183	18 (9.8%)
Patent	269	272	272	28 (10.3%)	266	53 (19.9%)
Chem Journal	335	342	341	26 (7.6%)	320	58 (18.1%)

Table 2. Gold standard data for the acronym identification and resolution tasks

3.3.2 Acronym Identification

Algorithm: We ran the algorithm on the 150 documents, which output the unique acronym identified and their resolutions. Resolutions not found are marked 'unknown.' For example, common acronym, such as *PM* to represent *afternoon*, are rarely expanded in documents. Therefore, we can track errors in terms of not identify-

ing strings that are acronyms (task 1). We can also identify errors in resolution (task 2). Figure 1 provides the pseudocode used by the algorithm.

Crowdsourcing: We had workers on Amazon Mechanical Turk (MTurk) read each document and identify the acronyms (task 1). Figure 2, left, provides a screenshot. Each document was processed by nine workers; we tracked the order in which they evaluated the documents. To determine how the number of crowdworkers affects quality, we calculated consensus judgments as follows. We took the first three workers submitting results as a set and evaluated majority consensus (2 of 3) and strong majority (3 of 3) consensus decisions. We repeated this majority and strong majority assessments for sets of the first 5, 7, and 9 workers making submissions. Each worker was paid $0.03 per document. For each, we evaluated recall, precision, and accuracy calculated against our gold standard.

To reduce noise, we provided "honeypots," where acronyms were provided in the document and could be found easily. Workers who did not identify these correctly had their answers removed from the pool, and the task was relisted.

3.3.3 Acronym Resolution

Algorithm: Acronym resolution was done in the same process as identification. The analysis was limited to the gold standard acronyms identified (e.g., 183 for the News collection). We found that expanding the window size used by Dannélls did not

Short-form acronym detection (the identification step):
Break document into terms and loop through each term;
IF (the string's length is between 2 and 10 characters) **AND** (the term contains a string of alphabetic, numeric and special characters such as '-' and '/') **AND** (it contains at least two uppercase letters) **AND** (it does not match terms in a stop list) **THEN**
Identify the string as a short-form acronym;
End;

For long-form acronym detection (the resolution step):
Examine a term window of size *min*(|A|+5, |A|*2)), where |A| is the short-form acronym length, to the left and to the right of the short-form acronym;
IF (all the terms within the term window are contained within parentheses) **OR** (the first letter of a term within the term window matches the first letter of the short form acronym) **THEN**
Identify the string that begins with that term as a potential candidate for a long form of that acronym;
End;

For all long-form acronym candidates:
IF (the first letter of two consecutive words in the string matches the first letter in two or more consecutive capital letters of the short-form acronym) **AND** (the string does not contain a colon, semi-colon, question mark or exclamation mark) **AND** (the maximum length of the string is *min*(|A|+5,|A|*2), where |A| is the sort-form acronym length)
AND (the string doesn't contain only upper-case letters) **THEN**
Identify the shortest string of terms as our long-form acronym description;
End;

Fig. 1. Pseudocode for the algorithm used for acronym detection and resolution.

increase the algorithm's accuracy. Figure 1 provides the pseudocode used by the algorithm.

Crowdsourcing: In a second crowdsourcing run we gave MTurk workers the same documents but with the gold standard acronyms highlighted and asked to resolve them. Figure 2, right, provides a screenshot. Each worker was paid $0.10 per document. Again, we had nine workers process each document and repeated our assessment of majority consensus and strong majority consensus for the first 3, 5, 7, and 9 workers to make submissions. Also, we asked each worker to mark whether they used the information solely from the document, or they used common knowledge / outside information. As with the acronym identification task, we provided crowdworkers with short-form acronyms where the resolution was provided in the document as "honeypots." Workers who were unable to identify these had their results removed from the pool and the task was relisted.

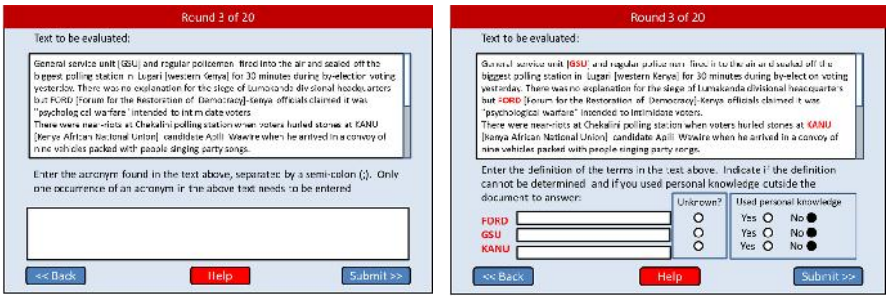

Fig. 2. Screenshots of the acronym identification (left) and resolution (right) tasks provided to crowdworkers

Game: The game was designed to provide players with a more entertaining and challenging method of resolving acronyms. Figure 3, left, shows a screenshot. Using the same highlighted documents players had to resolve the terms within a specified time limit (2 minutes per document) and were given real-time accuracy scores (Figure 3, right). A leaderboard was provided for top scorers to enter their names. We listed the game on MTurk and compensated each worker $0.05 to start the game and encouraged them to continue playing (and resolving acronyms) for as long as possible.

To accommodate acronym resolutions that were similar but not identical to the gold standard, we stemmed the answer provided and the gold standard definition and did a simple character string match on the stemmed terms (e.g., *South African Defence Forces* and *South African Defence Force*, when stemmed, both match the long form of *SADF*). If they matched, we increased the player's score. Although this only required for a few participants, later examination showed this technique was 96% accurate at correctly assessing the player's answer. Accuracy was determined as the proportion of acronyms in our gold standard that were correctly resolved.

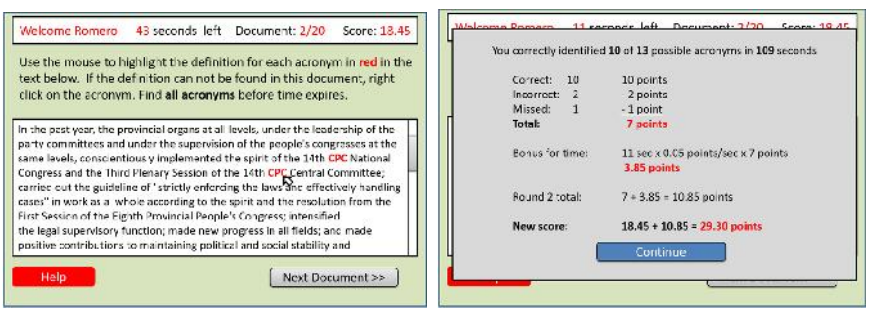

Fig. 3. Screenshots of the acronym resolution task (left) and the scoring calculations (right) provided to game participants

5 Results and Discussion

5.1 Acronym Identification

Table 3 shows the number of acronyms identified by the crowd and machine algorithm for nine assessors.

Text Collection	Gold Std (GS)	By Crowd		By Machine Algorithm		
		# Retrieved (5/9 majority)	# Correct (5/9 majority)	# Retrieved	# Correct	# Agreeing w/ Crowd
News	185	198	182	175	144	140
Patent	272	276	259	214	166	162
Chem Journal	341	344	335	295	239	242

Table 3. Acronym identification by the crowd and the machine algorithm

Table 4 reports accuracy, recall, and precision using majority score for crowdsourcing workers. Comparing these with scores obtained by the algorithm, the difference for each collection is significant, $F(2,147) = 4.32$, $p = 0.015$. We observe that the crowd identified more acronyms than the algorithm in all three collections with greater recall, precision and accuracy scores

		News			Patent			Chemistry Journal		
		A	P	R	A	P	R	A	P	R
Crowd	Majority (2/3)	0.959	0.968	0.989	0.952	0.996	0.952	0.994	0.994	0.982
	Strong Majority (3/3)	0.953	0.989	0.984	0.949	1.000	0.952	0.991	0.997	0.982
	Majority (3/5)	0.939	0.989	0.984	0.942	1.000	0.952	0.982	1.000	0.982
	Strong Majority (4/5)	0.939	0.989	0.984	0.942	1.000	0.952	0.980	1.000	0.982
	Majority (4/7)	0.934	0.989	0.984	0.942	1.000	0.952	0.974	1.000	0.982
	Strong Majority (6/7)	0.926	1.000	0.984	0.938	1.000	0.952	0.974	1.000	0.982
	Majority (5/9)	0.919	1.000	0.984	0.938	1.000	0.952	0.974	1.000	0.982
	Strong Majority (7/9)	0.919	1.000	0.984	0.938	1.000	0.952	0.974	1.000	0.982
Machine Algorithm		0.823	0.823	0.778	0.776	0.776	0.610	0.810	0.810	0.701

Table 4. Accuracy (A), precision (P) and recall (R) for the acronym identification task

5.2 Acronym Resolution

Table 5 provides accuracy scores for the three methods (algorithm, crowdsourcing, and games). Again, we see that the human computation methods, irrespective of

worker set size, do better than the algorithm. Although the crowd format slightly outperformed the game variant; a one-way between-subjects ANOVA found no significant difference in accuracy between them. We believe the lower performance of the game, although slight, may be due to distractions introduced by the game. There was a significant difference between the crowd/game formats and the machine format.

		Document + Ext. Knowledge			From Document Only		
		News	Patent	Journal	News	Patent	Journal
Crowd	Majority (2/3)	**0.984**	0.941	**0.979**	**0.876**	**0.732**	**0.806**
	Strong Majority (3/3)	0.973	0.923	0.968	0.870	0.721	0.798
	Majority (3/5)	0.968	0.923	0.971	0.865	0.721	0.801
	Strong Majority (4/5)	0.968	0.912	0.959	0.859	0.713	0.792
	Majority (4/7)	0.957	0.912	0.962	0.859	0.713	0.792
	Strong Majority (6/7)	0.946	0.904	0.956	0.854	0.710	0.789
	Majority (5/9)	0.951	0.904	0.956	0.854	0.710	0.789
	Strong Majority (7/9)	0.941	0.904	0.956	0.849	0.710	0.789
Game	Majority (2/3)	0.951	**0.949**	0.977	0.849	0.724	0.798
	Strong Majority (3/3)	0.930	0.934	0.959	0.832	0.717	0.786
	Majority (3/5)	0.930	0.938	0.959	0.832	0.721	0.786
	Strong Majority (4/5)	0.908	0.919	0.950	0.816	0.710	0.783
	Majority (4/7)	0.930	0.919	0.979	0.816	0.710	0.783
	Strong Majority (6/7)	0.908	0.915	0.974	0.811	0.706	0.780
	Majority (5/9)	0.908	0.915	0.974	0.811	0.706	0.780
	Strong Majority (7/9)	0.908	0.915	0.971	0.811	0.706	0.780
Machine Algorithm		0.778	0.610	0.701	0.778	0.610	0.701
Increase in score by best human computation method over machine algorithm:		0.205	0.338	0.279	0.097	0.122	0.105
ANOVA	F-value	16.81	63.85	217.85	5.28	23.19	22.45
	Significance	0.056	*0.015*	*0.005*	0.159	*0.041*	*0.043*

Table 5. Table 5: Accuracies for the acronym resolution task

Post-hoc comparisons using the Bonferroni test indicated significant differences between the crowd/game and the machine for the patent and chemical journal collections. Thus, for these collections, the crowd and game formats were both able to outperform the machine, and this difference was significant for the two collections even

when limiting information used to the document. This indicates that human computation methods are better at pulling definitions from the text even without relying on outside information (Q4). The patterns of gain are similar for games with respect to the machine approach. The resolution of acronyms benefits from human computation methods most when documents are more challenging to read, implying the resolution of acronyms is more difficult as well (Q2). We also note that the algorithm performs reasonably well in these domains though possibly not as well in the biomedical domains for which it was developed (Q1).

5.3 Number of Workers

To answer our research question on the number of workers needed (Q3), we compare the performance scores obtained using all nine assessors with scores using only the first three assessors. In both tasks, we found no significant difference in the quality of results. This demonstrates that we need only a few assessors from the crowd to obtain quality results. This is similar to the findings by Snow et al. on a separate study of non-experts in NLP tasks [14]. Cutting the number of assessors from 9 to 3 can reduce assessment costs by two thirds. This, coupled with no measurable gain in quality reinforces our reluctance to use more than three members of the crowd (or game participants) in similar identification and resolution tasks.

Using a paired t-test, we also found no significant difference in quality between the crowd and game approaches. For the acronym resolution task, we spent $135.00 for the nine crowd assessors and half that amount to obtain similar quality in the game with a similar number of assessors. This does not consider the fixed cost required for game development – for short term evaluation; this cost is a factor; for an evaluation over a much longer period, it becomes a trivial cost.

Some acronyms, particularly those in patents, could not be ascertained in the text by any approach and required background knowledge of the domain. The largest reason for acronyms being missed by humans in the identification task is that people confused abbreviations (e.g., Sr. for senior) with acronyms. Machine-based techniques rarely make this type of mistake. Overall, humans were able to resolve 93%, 86% and 96% of the acronym identification errors made by the machine approach and 44%, 22% and 33% of the acronym resolution errors (on acronyms not requiring external knowledge) made by the machine approach for the news, patent, and chemical journal collections, respectively (Q5). We believe this may be due to the machine algorithm evaluated rules on acronym identification without flexibility, whereas humans were flexible and resolved acronyms that did not follow common rules (such as *ATM machine*, where *machine* confounds the algorithm's ability to resolve the long-form equivalent). Humans provided value when there were several candidate definitions, and the correct one required context to resolve, such as resolving *PM* when both *Post Meridian* and *Prime Minister* appear in the document. Humans also provided value in acronym resolution when definitions in the text contained embedded short-form acronyms (e.g., *PRESTO* is defined as *Precursory Research for Embryonic S&T Program*, and *S&T* is defined elsewhere in the same document as *Science and Technology*).

6 Conclusion and perspectives

We have applied two separate approaches (a machine algorithm and a crowdsourcing approach) to three different publicly-available text collections in an algorithm identification task. We applied these two approaches, plus a game-based approach, to a task to find the long-form acronym definition to the short-form acronym in the same text collections. The machine algorithm we used, designed for biomedical text, was unable to obtain the same accuracy, precision and recall rates for any of our three text collections. We evaluated that an increase in accuracy of 10-30% can occur when non-expert human computation methods are used. For those acronyms incorrectly resolved by the algorithm, we found that many of these errors did not rely on external information from the human participant (e.g., using *ETA* for *Estimated Time of Arrival*); although external knowledge plus the ability for humans to resolve acronyms with the information in the text provided the best results.

For acronym identification and resolution tasks, we found adding additional assessors did not provide an improvement in accuracy, precision or recall. We also found that most algorithmic errors were a result of the inability to evaluate exceptions to specified rules, which humans can do relatively well. In the horizon, more advanced techniques for identifying and resolving acronyms may be able to mimic human decision-making, closing the gap between human computation and machine approaches.

It should be noted that the costs were initially much higher to set up the gamification interface over a standard interface on a crowdsourcing platform like MTurk. However, because we paid a flat fee to game participants, making the game "sticky," or capable of captivating a user's attention for a sustained period of time, gamification can save money in the long run, since people keep participating as long as their interest is maintained. Adding stickiness to the game interface had the adverse effect of making it more complex. This may have served as a distraction from the task of resolving acronyms (as observed from the game's slightly lower accuracy scores when compared to the crowd interface), mitigating some of the gamification interface's potential. We, therefore, believe that the crowd interface provides the best overall approach across each collection for acronym identification and resolution.

In future work, we plan to look at low-resource languages, particularly those that do not use capital letters. We also plan to incorporate phonotactics; short forms with no vowel or "y" are more likely to be acronyms (since they are less likely to be real words); the same goes for consonant combinations that are not normally found in the English language. One particularly tricky area considered for a follow-up study is acronyms composed of initial syllables of words. These tend to be phonotactically well-formed words and often become so lexicalized that people no longer realize they are acronyms. *Scuba* and *laser* in English, *nazi* in German and *kolkhoz* in Russian are of this type.

References

1. Michael Molloy. 2010. AcronymFinder.com passes the 1 million definition milestone. Retrieved from http://blog.acronymfinder.com/2010/12/acronymfindercom-passes-1-million.html
2. William R. Hersh and Ravi Teja Bhupatiraju. 2003. TREC genomics track overview. *TREC, vol. 2003*, pp. 14-23.

3. Sungrim Moon, Bridget McInnes, and Genevieve B. Melton. 2015. Challenges and practical approaches with word sense disambiguation of acronyms and abbreviations in the clinical domain. *Healthcare informatics research* 21, no. 1 (2015): pp. 35-42.
4. David Sánchez and David Isern. 2011. Automatic extraction of acronym definitions from the Web. *Applied Intelligence* 34(2) pp. 311-327. doi: 10.1007/s10489-009-0197-4
5. R. Menaha, M.Barkavi , P.Guha Prashanthini , R.Narmadha. 2016. A web based approach: Acronym Definition Extraction. *International Research Journal of Engineering and Technology (IRJET)* 3(2) pp. 1199-1206.
6. Michael R. Glass, Mohammad Faisal Mahbub Chowdhury, and Alfio M. Gliozzo. 2017. Language Independent Acquisition of Abbreviations. arXiv preprint arXiv:1709.08074.
7. Katrin Kirchhoff and Anne M. Turner. 2016. Unsupervised Resolution of Acronyms and Abbreviations in Nursing Notes Using Document-Level Context Models. In Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis (pp. 52-60).
8. Ariel S. Schwartz and Marti A. Hearst. 2002. A simple algorithm for identifying abbreviation definitions in biomedical text. *Biocomputing*. pp. 451-462.
9. Dana Dannélls. 2006. Automatic acronym recognition. *Proceedings of the Eleventh Conference of the European Chapter of the Association for Computational Linguistics: Posters & Demonstrations*. Association for Computational Linguistics.
10. Tim Finin, Will Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in Twitter data with crowdsourcing. *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*. Association for Computational Linguistics.
11. Ellen Voorhees and Donna Harman. 2000 Overview of the sixth text retrieval conference (TREC-6)." Information Processing & Management 36(1) pp. 3-35.
12. Mihai Lupu, Florina Piroi, Xiangji Huang, Jianhan Zhu, and John Tait. 2009. Overview of the TREC 2009 chemical IR track. York University Downsview (Ontario).
13. Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. 2008. Cheap and fast---but is it good?: evaluating non-expert annotations for natural language tasks *Proceedings of the conference on empirical methods in natural language processing*. pp. 254-263. Association for Computational Linguistics.

Appendix A
Perl program used for machine algorithm

```
#!/usr/bin/perl
use File::Find;
use Lingua::EN::Sentence qw(get_sentences);
use Cwd('abs_path');
use Data::Dumper;
use strict;

my $top_dir = abs_path($ARGV[0]);
my % stop_list_term = ('a' => 1, 'on' => 1, 'of' => 1,
'in' => 1, 'the' => 1, 'at' => 1, 'an' => 1, 'for' =>
1,);
my $stop_list_regexp = join('|', keys % stop_list_term);
$stop_list_regexp = qr/\b$stop_list_regexp\b/;
my %acronyms;

sub get_acronym {
    my($s) = @_;
    my $i = -1;
    my @acrs = ();
    foreach my $w((split(/\s+/, $s))) {
        $i++;
        my $nxt;
        next
        if ($w =~ /[:;!?]/);
        foreach my $stop (keys %stop_list_term){
            if ($w =~ /^$stop$/i){
                $nxt=1;
                last;
            }
        }
        next if ($nxt);
    my $weight = $# {[($w =~ /[[:upper:]]/g)]} +1;
        $w =~ s/([(\[])?"?(\w+)("?[)\]][\., ]?)?/$2/g;
        if ($weight>= 2 && length $w >= 2 && length $w <=
10 && $w =~ /^[a-z0-9\-\/']+$/i){
            $w =~ s/^(.+)'\w$/$1/;
            my $acr = {};
            $acr->{word} = $w;
            $acr->{index} = $i;
            $acr->{weight} = $weight;
            @{$acr->{letters}} = ();
            foreach ( split(/([[:upper:]])/, $w)) {
                if ($_) { push @{$acr->{letters}}, lc $_;
}
            }
```

```
            push @acrs, $acr;
        }
    }
    return \@acrs;
}

sub is_start_word_as_acr {
    my ($word, $acr) = @_;
    my $fl = substr($word, 0, 1);
    if (lc $fl eq lc substr($acr, 0,1)){
        return 1;
    }
    return 0;
}

sub check_for_lf {
    my ($s, $acr, $docno) = @_;
    my @result;
    my $term_window = ((length $acr->{word})+5 < (length
$acr->{word})*2)?
            ((length $acr->{word})+5):(length $acr-
>{word})*2;
    my@ words = split(/\s+/, $s);
    if (join(' ', @words[$acr->{index}+1 .. $#words])
=~/^[(\[]"?([^)\]]+)"?[)\]]/){
        push @{$acronyms {$docno}->{$acr->{word}}}, $1;
    }
    my @words = map {s/([(\[])?(\w+)([)\]])?/$2/; $_}
@words;
    my $left_boundary = ($acr->{index}-$term_window >=
0)?($acr->{index}-$term_window):0;
    my $right_boundary = ($acr->{index}+$term_window <=
$#words)?($acr->{index}+$term_window):$#words;
    my@ ls_lf = @words[$left_boundary .. $acr->{index}-
1];
    my@ rs_lf = @words[$acr-
>{index}+1 .. $right_boundary];
    foreach(@ls_lf){
        if(is_start_word_as_acr($_, $acr->{word})){
            push @result, [@ls_lf];
            last;
        }
    }
    foreach(@rs_lf) {
        if (is_start_word_as_acr($_, $acr->{word})) {
            push @result, [@rs_lf];
            last;
        }
```

```
    }
    return \@ result;
}

sub some_checks_for_description {
    my ($candidate, $acr, my $fl_acr) = @_;
    if( $candidate = ~/[:;?!,]/ or $candi-
date !~ /[[:lower:]]/
            or $candidate =~ $acr->{word}or $candidate =
~/^\s+$/) {
        return 0;
    }
    if (length $acr->{word}-$fl_acr >= 2 ){
        return 0;
    }
    ## print "=xx ", (grep {!/$stop_list_regexp$/}
split(/ /, $candidate)), "\n";

    if( scalar ( grep {!/^$stop_list_regexp$/} split(/ /,
$candidate)) <2){
        return 0;
    }
    my %tmp = ();
    foreach (split(/ /, $candidate)){
        if(exists $tmp {$_}) {
            return 0;
        }else{
            $tmp {$_}=1;
        }
    }
    return 1;
}

sub get_description {
    my($acr, $lf) = @_;
    my@ candidates = ();
    foreach my $longform (@$lf) {
        my @fl_acr;
        my %w_for_skip = ();
        my($acr_l, $_exit, $candidate, $last_term) = ({},
0, '', '');
        my@ tmp_candidates = ();
        while (!$_exit) {
            my ($skip, $start, $check) = (0,0,0);
            $candidate = '';
            @fl_acr = map {($_ = ~/[A-
Z]/)?({lc $_=>1}):({$_=>0})} split( //, $acr->{word});
            my $acr_l = shift @fl_acr;
```

```
            while ((values %$acr_l)[0] == 0) {
                    $acr_l = shift @fl_acr;
                }
                foreach my $lf_term (@$longform) {
                    # # print$acr->{word}, "
$lf_term \n";
                    next if(exists
$w_for_skip {$lf_term});
                    if (!exists $acr_l-
>{(lc substr($lf_term, 0, 1))} && $start && $skip == 0 &&
$#fl_acr >= 0 && !exists $stop_list_term {$lf_term}) {
                        # # print "candidate = $candi-
date\n";
                        $w_for_skip {$last_term}=1;
                        if
(some_checks_for_description($candidate, $acr,
$#fl_acr)){
                            push @tmp_candidates, $candi-
date;
                            # # printDump-
er \@candidates;
                        }
                        $check = 1;
                        last;
                    }
                    $last_term = $lf_term;

                    if (exists $acr_l-
>{(lc substr($lf_term, 0, 1))} &&
                            $acr_l->{(lc substr($lf_term,
0, 1))} == 1) {
                        if ($skip) {$skip=0;}
                        unless($start) {$start=1;}
                        # print "add $lf_term \n";
                        $candidate. = "$lf_term ";
                        # print $candidate, "\n";
                        $acr_l = shift @fl_acr;
                        next;
                    }
                    if ((exists $acr_l-
>{(lc substr($lf_term, 0, 1))} && $start &&
                            $acr_l->{(lc substr($lf_term,
0, 1))} == 0) || $skip == 1){
                        unless($skip) {$skip=1;}
                        # print "add1 $lf_term \n";
                        $candidate .= "$lf_term ";
                    }
```

```
                    if (exists $stop_list_term{$lf_term}
&& $start && $#fl_acr >= 0){
                        # print "add2 $lf_term \n";
                        $candidate .= "$lf_term ";
                    }
                }
                if ($check == 0) {
                    $_exit = 1;
                }
            }
            if (some_checks_for_description($candidate,
$acr, $#fl_acr)) {
                push @candidates, $candidate;
                # # printDumper \@candidates;
            }elsif($#tmp_candidates >= 0) {
                push @candidates, pop @tmp_candidates;
            }
        }
        return \@candidates;
    }

    sub acronym {
        my ($docno, $lines) = @_;
        $lines = ~s/\n\n/\.\n/mg;
        my $sentences = get_sentences($lines);
        foreach my $s (@$sentences){
            if ($s = ~/--/) {
                $s = (split(/--/, $s))[1];
            }
            next if $#{[($s = ~/[[:lower:]]/g)]} == -1;
            my $acrs = get_acronym($s);

            foreach my $acr (@$acrs) {
                my $lf_for_check = check_for_lf($s, $acr,
$docno);
                my $descr;
                if ($acr->{word} && $#{$lf_for_check}>-1)
{
                    $descr = get_description($acr,
$lf_for_check);
                }else{
                    push @{$acronyms{$docno}->{$acr-
>{word}}}, 'undefined';
                    next;
                }
                if ($#{$descr} >= 0) {
                    foreach(@$descr) {
```

```
                        push @{$acronyms {$docno}->{$acr-
>{word}}}, $_;
                    }
                }else{
                    push @{$acronyms{$docno}->{$acr-
>{word}}}, 'undefined';
                }
            }
        }
    }

    sub wanted {
        my $fn = $File::Find::name;
        my($doc, $docno, $lines);
        if (open(F, $fn)) {
            while (my $str = <F>){
                if
($str =~ m#<DOCNO>\s?(\S+)\s?</DOCNO>#){
                    $docno = $1;
                }
                if ($str =~ /<TEXT>/) { $doc = 1; }
                if ($str =~ /(<\/TEXT>|<\/DOC>)/){
                    $doc = 0;
                    if ($docno) {
                        if ($lines =~/\w+/) {
                            acronym($docno,$lines);
                        }
                        $docno = 0;
                    }
                    $lines='';
                }
                if ($doc) {
                    if ($str =~/<[^>]+>.+<[^>]+>/){
                        next;
                    }
                    $lines .= $str;
                }
            }
            close F;
        }
    }

    find(\&wanted, $top_dir);

    foreach my $docno (keys %acronyms){
        foreach my $acr(keys %{$acronyms{$docno}}){
            my $defined = 0;
```

```
            foreach my $descr (@{$acronyms{$docno}-
>{$acr}}){
                if ($descr ne 'undefined') { $defined =
1; }
            }
            my % tmp = ();
            unless($defined) {
                print "$docno $acr - undefined\n";
                next;
            }
            foreach my $descr (@{$acronyms {$docno}-
>{$acr}}){
                $descr =~s/\s$//g;
                if ($defined && $descr ne 'undefined') {
                    unless(exists $tmp{$descr}) {
                        print "$docno $acr - $descr\n";
                        $tmp{$descr}=1;
                    }
                }
            }
        }
    }
```