**Noname manuscript No.**
(will be inserted by the editor)

# Acronyms: Identification, Expansion and Disambiguation

**Kayla Jacobs · Alon Itai · Shuly Wintner**

**Abstract** Acronyms—words formed from the initial letters of a phrase—are important for various natural language processing applications, including information retrieval and machine translation. While hand-crafted acronym dictionaries exist, they are limited and require frequent updates. We present a new machine-learning-based approach to automatically build an acronym dictionary from unannotated texts. This is the first such technique that specifically handles *non-local acronyms*, i.e., that can determine an acronym's expansion even when the expansion does not appear in the same document as the acronym. Our approach automatically enhances the dictionary with contextual information to help address the acronym disambiguation task (selecting the most appropriate expansion for a given acronym in context), outperforming dictionaries built using prior techniques. We apply the approach to Modern Hebrew, a language with a long tradition of using acronyms, in which the productive morphology and unique orthography adds to the complexity of the problem.

## 1 Introduction

An **acronym** is a word created from the initial components of a phrase or name, called the **expansion**. For example, *CIA* is an acronym with the expansion *Central Intelligence Agency*, though it has additional expansion possibilities depending on context, such as *Culinary Institute of America* or *Certified Internal Auditor*.

Understanding the relationship between acronyms and their expansions is important for several natural language applications, including:

K. Jacobs
Computer Science Department, Technion, Haifa 32000, Israel
E-mail: kayla@cs.technion.ac.il

A. Itai
Computer Science Department, Technion, Haifa 32000, Israel
E-mail: itai@cs.technion.ac.il

S. Wintner (corresponding author)
Department of Computer Science, University of Haifa, Haifa 31905, Israel
E-mail: shuly@cs.haifa.ac.il

| | |
|---|---|
| Information retrieval | When searching for a document, a query containing an acronym should also return documents containing the acronym's expansion—and vice versa. |
| Machine translation | Acronyms often present a challenge when automatically translating text from one language to another. If the source text includes an acronym, it is rarely sufficient to simply transliterate the acronym letters; indeed, the acronym may not even exist in both languages. |
| Text understanding | An acronym in a text may not be familiar to the reader (whether human or computer), or it may have additional expansions beyond the intended one, each of which can change the interpretation of the text. Recognizing the correct meaning of an acronym in context is critical to understanding. |
| Text summarization | Using an acronym instead of its expansion can improve text summarization. |

Typically, processing tools rely on acronym dictionaries containing acronyms and their expansions. However, the collection of acronyms is an open set, with new acronyms and new expansions constantly being added for company and organization names, technical terms, etc. [19]. These dictionaries are thus far from complete and require frequent updates.

Most of the dominant existing approaches to identifying acronyms and expansions in free text focus on **local acronyms**, whose expansions appear in the same document, typically in the same sentence or nearby sentence and frequently related by parentheses. For example, the following sentences all contain the local acronym *CIA* with various expansions.

1. *The CIA (Central Intelligence Agency) released its budget.*
2. *The chef trained at the Culinary Institute of America (CIA).*
3. *Her job title is Certified Internal Auditor, or CIA for short.*
4. *I love the Cleveland Institute of Art. I'm even a registered CIA member.*

In contrast, **non-local (global) acronyms** are unaccompanied by their expansion in the same document, written with the (not necessarily correct) assumption that the reader is already familiar with the acronyms' intended meanings. Non-local acronyms are more challenging to interpret since the expansion cannot be found nearby. To assess the prevalence of non-local acronyms we conducted an evaluation on a subset of the Hebrew Wikipedia corpus.[1] Out of 1509 acronyms, only 520 were immediately followed or preceded by expansions involving parentheses as in (1) and (2) above; of those, only 85 obeyed the formation rules of Hebrew acronyms (see Section 3.4). Manual inspection revealed that only 71 of them, or 4.7%, were indeed expansions. In other words, one can only count on fewer than 5% of the acronyms to be local.

The main contribution of this paper is a machine-learning-based method that automatically extracts acronyms and their expansions from a text corpus, thereby producing an easily-updatable acronym-expansion dictionary. The approach specifically includes non-local acronyms, making it the first work, to our knowledge,

[1] `http://www.mila.cs.technion.ac.il/eng/resources_corpora_wikipedia_2013.html`

to address this important acronym class. The resulting dictionary is also automatically enhanced with contextual information that helps during acronym disambiguation (selecting the expansion most appropriate for a given acronym in context). We implemented the technique on Modern Hebrew texts, but it can be applied to other languages and to specialized domains (Section 7).

## 2 Previous Work

A simple approach for building a dictionary of local acronyms from text, by assuming that either the acronym or the expansion is written within parentheses, like in (1) or (2) above, was suggested by [29]. This algorithm was expanded and applied to Swedish biomedical texts [5,6], in one of the few works addressing non-English acronyms. Pattern-based rules for English acronyms were also described by [25], who identified expansions using text markers, such as parentheses and cue words like *for short* in (3) above. A more sophisticated approach using regular expressions to recognize English acronyms and an acronym-expansion letter-matching algorithm was developed by [17].

A few works focused on extracting acronyms and their expansions from sources other than plain-text documents. These include an analysis of web page source code, looking for HTML tags that included both an acronym and its possible expansion, such as `<a name="CSS">Cascading Style Sheet</a>` [34]. Another approach used web search query logs [16], looking for consecutive queries by the same user in which first an acronym was searched for, then its (possible) expansion, following a presumed failure of the first search query to return the desired results. Such approaches are naturally very limited in their scope.

Several studies [35,23,33,7] addressed the issue of matching and ranking potential acronym-expansion pairs once they are identified, using machine learning with linguistically-informed features to classify pairs as related or not. Specifically, this task can be viewed as a word sense disambiguation task, especially in biomedical domains [31,18,24]. We employed a similar approach (see Section 5.3), albeit with new and more powerful features.

Acronyms in Hebrew have primarily been studied through a qualitative linguistic or historical lens [32,28,30,22]. There are, however, a few computational studies [8,9,10,11], in which Hebrew and Aramaic acronym disambiguation systems were developed for classical Jewish texts (primarily in pre-Modern Hebrew), using both existing manually-crafted acronym-expansion dictionaries and stochastic approaches. Manual acronym disambiguation in this genre was shown to be a time-consuming and difficult task for human annotators; even highly-trained domain experts, given multiple-choice options which always included the correct answer, experienced difficulties [12]. Machine learning approaches, in contrast, proved highly accurate [11].

## 3 Acronyms in Hebrew

Hebrew acronyms have unique linguistic features, some aiding our research goals but others presenting extra challenges. When relevant, we provide comparisons to

English, the language of most prior research on acronyms. In Section 7 we discuss how to adapt our techniques to other languages.

### 3.1 History and Prevalence

While acronyms are a relatively recent addition to the English language, first significantly appearing in the 20th century [19], Hebrew has a long history of acronyms, dating back to at least the first century CE [28]. Acronyms are especially frequent in the specialized genres of Jewish religious and legal writings of all historical periods [10] and in modern Israeli military texts [28].

In the secular Modern Hebrew texts that we investigated (presented in Section 4), acronyms account for about 1% of the tokens and 3% of the types, appearing more frequently in news and encyclopedia genres than in literature.

Hebrew processing poses various challenges [15], including the complexity introduced by its complicated morphology and orthography, particularly prefixed function words (explained in Section 3.3); the general paucity of language processing resources; and common acronym formation rules that involve multiple initial letters from expansion words (see Section 3.4).

### 3.2 Orthographic Styling

English acronyms are written in a wide variety of capitalization and punctuation styles, such as *M.S./MS/M.Sc./MSc/MSC* = *Master of Science*, *au* = *atomic unit*, and *3-D/3D* = *3-dimensional*. This diversity of representations makes identifying English acronyms a non-trivial problem, especially because an acronym may appear in the same style as an ordinary word.

In contrast, Hebrew acronyms are easy to identify. They are almost always written as strings of two or more Hebrew letters,[2] with an internal double-quote mark (`"`) located before the last letter. For example, `mnk"l` is a Hebrew acronym with the expansion `mnhl klli` (*manager general*, *"chief executive officer (CEO)"*).

### 3.3 Function Words, Prefixes and Suffixes

Each word in an expansion usually contributes at least one letter to the acronym. A major exception are function words like *the, of*, and *to* in English; and `lmyn` (*for*) and `el` (*of*) in Hebrew. These function words are often entirely skipped when forming acronyms; for example, *The Association of Americans and Canadians in Israel* is represented as *AACI*, not *TAOAACII*.

In English, function words are always separated from other words by spaces and thus are easy to recognize. In Hebrew, however, many are orthographically represented as prefixes: `b+` (*in/on*), `h+` (*the*), `w+` (*and*), `k+` (*as*), `l+` (*to/for*), `m+` (*from*), and `e+` (*that*). Certain combinations of these prefixes are also possible, such as

[2] To facilitate readibility, we use the MILA Transliteration Scheme [15], a left-to-right Roman character transliteration of Hebrew letters. In lexicographic order, the alphabet is `abgdhwzxviklmnsypcqret`.

mh+ (*from the*) or wke+ (*and when*) (there are approximately 100 such combinations in common use). Prefixes preceded slightly more than half (51%) of acronym tokens that we manually analyzed. Within expansions, these prefixed particles may or may not legitimately contribute to the acronym letters, sometimes even *instead* of the content words they precede.

Another problem with prefix function words is the danger of misidentifying them as non-prefixed initial letters of acronyms. For example, the form mnk"l = mnhl klli (*manager general*, *"CEO"*) can be mistakenly interpreted as m+nk"l, where m+ (*from*) is interpreted as a prefix, attached to the remaining acronym nk"l. This "prefix or not?" problem is a general issue with all Hebrew words, not just acronyms. Typically morphological analyzers use a lexicon of known word types to check whether the first letter (or letters) can serve as prefixes, but this approach is limited when applied to acronyms, as many are not in the lexicon [15].

Hebrew acronyms, like many words in the language, can have a variety of suffixes as well. Inflectional suffixes mark number and gender, pronomial suffixes function as shortened forms of possessive personal pronouns, and derivational suffixes can change lexical category (e.g., noun to adjective). Independent of the type of Hebrew suffix, the double-quote mark still appears before the last letter of the acronym's *base* form. For example, applying the feminizing inflectional suffix +it to mnk"l = mnhl klli (*manager general*, *"CEO"*) results in mnk"lit, not mnkli"t. Therefore, unlike in the case of prefixes, it is simple to identify which part (if any) of an acronym is a suffix.

### 3.4 Formation Rules

An acronym is formed from its expansion by concatenating certain letters from the expansion words. The pattern of which letters are chosen is the **formation rule** for the acronym-expansion pair, and there is a strong preference for initial letter(s) of expansion words. The most common formation rule in both English and Hebrew takes the very first letter of each word in the expansion, such as aa"k = ala am kn (*but if thus*, *"unless"*). However, we found that in about half of all Hebrew acronym-expansion pairs, at least one of the expansion's words contributes more than a single letter. For example, in kdwh"a = kdwr harc (*sphere-of the+land*, *"Earth/globe"*), the first expansion word contributes the first three letters, and the second word contributes the last two.

We introduce a notation for representing the formation rules in square brackets, with numbers denoting the position of the letter(s) of the word that appear in the acronym, and with words separated by commas. For example, [1, 1, 1] and [123, 12] denote the respective formation rules for the examples above.

To identify the formation rules that relate acronyms and their expansions, we developed a letter-matching algorithm that, given an acronym and its expansion, outputs the formation rule(s) that relate the two. We ran this algorithm on a gold-standard set of known acronym-expansion pairs (described in Section 4); examples for the seven most common resulting rules are shown in Table 1, which together cover 92% of the set.

The algorithm works by matching acronym letters to the initial letters of the expansion words. It allows skipping initial letters that could be prefixes (explained in Section 3.3) as well as entire words. For example, bg"c = bit hmepv hgbwh lcdq

| Rule | Example | % |
|---|---|---|
| [1, 1] | x"k = xbr knst (*member-of parliament*, *"parliament member"*) | 43 |
| [1, 1, 1] | aa"k = ala am kn (*but if thus*, *"unless"*) | 15 |
| [12, 1] | mm"d = mmlkti dti (*governmental religious*, *"national religious"*) | 13 |
| [1, 12] | mw"m = mea wmtn (*give and+take*, *"negotiation"*) | 10 |
| [12, 12] | bim"e = bit mepv (*house-of trial*, *"court"*) | 6 |
| [123, 1] | swp"e = swp ebwy (*end-of week*, *"weekend"*) | 3 |
| [1, 1, 1, 1] | ayp"k = ap yl pi kn (*yet on as thus*, *"nevertheless"*) | 2 |

**Table 1** Examples for the most common Hebrew acronym formation rules, with their proportions in the gold-standard set (described in Section 4). Together, these seven rules cover 92% of the set.

(*house the+law the+high for+justice*, *"High Court of Justice"*) results in the formation rule [1, , h+2, l+2]. Here h+2 and l+2 indicate that the acronym was formed by skipping the prefix h+ in the third word and the prefix l+ in the fourth. Nonetheless, we found the incidence of prefix-skipping and word-skipping formation rules to be very rare.

When an acronym-expansion pair was related by more than one possible formation rule, we resolved the ambiguity by choosing the rule that minimized the number of skipped words and/or letters.

### 3.5 Hebrew Numbers: Gematria

One special class of Hebrew words that appear as "acronym-like" tokens are Hebrew numbers. The language has a system, called **gematria** (or **gimatria**), of representing numbers using the Hebrew alphabet, with each letter assigned a fixed numerical value. Historically, this provided a convenient way to represent numbers before the widespread adoption of the Arabic numeral system, and today's Modern Hebrew frequently uses this system for enumerating short lists (similar to the English practice of enumerating A, B, C, ...) and for denoting dates in the Hebrew calendar.

Numbers that cannot be represented by a single Hebrew letter alone are represented as the sum of the letters in a "word" written in the same orthographic style as acronyms. For example, i"b stands for 12, whereas te"x represents the number 708. Such forms are thus easily mistaken for true acronyms, though of course they do not have traditional expansions with matching letters. For example, k"a can represent the number 21 or the acronym k"a = kl axd (*every one*, *"everyone"*). We found that such forms comprise a non-negligible 16% of acronym-like types in Hebrew texts, enough to require special handling in our work.

## 4 Resources

*Corpus* We assembled a Modern Hebrew corpus of over 215,000 documents consisting of news articles, records of parliamentary proceedings, chapters of literary

books, and the text content of Hebrew Wikipedia.[3] In total, the size of the combined corpus was over 77 million Hebrew word tokens of nearly 100,000 types (not including numbers, punctuation, or non-Hebrew tokens).

*Gold-standard acronym dictionary* We curated a gold-standard collection of known acronym-expansion pairs collected from the union of three human-edited dictionaries. We discarded acronyms and expansions which appeared fewer than five times in the corpus, to ensure that the set was representative of the acronyms and expansions present in the corpus. We manually reviewed each of the remaining pairs to discard entries that were obviously typos or mistakes. The final high-quality set consisted of 885 acronym-expansion pairs. This set was used to identify the set of formation rules (Section 3.4), to train and intrinsically evaluate the dictionary-building classifier (Section 5.3.3), and to form a baseline dictionary for the acronym disambiguation evaluation (Section 6).

## 5 Building an Acronym Dictionary

As discussed in Section 2, prior methods of automatic dictionary-building from unstructured texts focused on local acronyms, whose expansions appear nearby in the same document. We developed a new approach which includes non-local acronyms, comprised of three steps: identifying acronyms (Section 5.1); identifying candidate expansions (Section 5.2); and matching acronyms and expansions (Section 5.3).

### 5.1 Identifying Acronyms

To extract the set of acronyms from the corpus, we took advantage of the unique orthography of Hebrew acronyms, described in Section 3.2. (Any adaptation of our methodology to other languages will likely be most challenging in this stage of the work, as discussed further in Section 7, since acronym-identifying rules are language-specific and typically more challenging than for Hebrew.) The result was 12,895 acronym types, from 766,074 acronym tokens. We removed suffixes (a trivial undertaking, as explained in Section 3.3) and discarded acronym types appearing fewer than five times in the corpus. The resulting set consisted of 3,862 acronym types covering 93% of all acronym tokens in the corpus.

### 5.2 Identifying Candidate Expansions

To identify candidate expansions, we first extracted all $n$-grams from the corpus, with $2 \leq n \leq 4$, since in the gold set, 97% of expansions were two to four words long. We discarded $n$-grams which were infrequent (fewer than five corpus instances), as well as those ending with a preposition or a quantifier, which indicated that the $n$-gram was an incomplete phrase and thus unlikely to be a

[3] All corpus materials were from MILA: Knowledge Center for Processing Hebrew [15], except the literary book chapters, which were generously provided by Justin Parry of the National Middle East Language Resource Center (NMELRC).

full expansion. For every remaining $n$-gram in the set, we next generated all the acronyms that it *could* form via the formation rules discovered in Section 3.4, and then filtered out infrequent acronyms (fewer than five corpus instances).

As an illustrative example, consider the 2-gram `bit xwlim` (*house-of sick-people*, *"hospital"*), which appeared 906 times in the corpus—well above the frequency threshold of five instances. Table 2 lists the most common formation rules for 2-grams, and the acronyms that result by applying these formation rules to `bit xwlim`, as well as their corpus frequencies.

| Rule | Acronym | Frequency |
|---|---|---|
| [1, 1] | `b"x` | 4 |
| [12, 1] | `bi"x` | 144 |
| [1, 12] | `bx"w` | 0 |
| [12, 12] | `bix"w` | 0 |
| [123, 1] | `bit"x` | 0 |
| [123, 12] | `bitx"w` | 0 |

**Table 2** All acronyms formable from the 2-gram `bit xwlim` (*house-of sick-people*, *"hospital"*), via each of the relevant common formation rules, and their corpus frequencies.

Only two of these acronyms actually appeared in the corpus, however, and only one (`bi"x`) appeared above the threshold collection frequency of five. Thus, this $n$-gram contributed one acronym/$n$-gram pair: `bi"x` $\stackrel{?}{=}$ `bit xwlim`.

For each acronym appearing at least five times in the corpus, we obtained all candidate expansions—collocations that yielded the acronym under one of the formation rules, also appearing at least five times in the corpus. Note that the criteria for being considered a candidate expansion are quite inclusive: unlike the `bit xwlim` example, most $n$-grams had many possible acronym matches; and similarly, most acronyms—especially the shorter ones—had many possible $n$-gram matches. For example, the acronym `bi"x` had 640 $n$-gram pairings, a few of which are shown in Table 3.

| Rule | $n$-gram |
|---|---|
| [1, 12] | `ba ixd`<br>(*come together*, *"come together"*) |
| [12, 1] | `bin xwwt`<br>(*between farms*, *"between farms"*) |
| [12, 1] | `bit xwlim`<br>(*house-of sick-people*, *"hospital"*) |

**Table 3** A few of the 640 candidate expansions for the acronym `bi"x`.

## 5.3 Matching Acronyms and Expansions

Armed with the list of acronyms and their candidate expansion $n$-grams, the next step was to determine which ones are likely to be actual expansions in certain con-

texts. Using standard machine learning techniques, we trained a binary classifier to predict, for a pairing of an acronym and an $n$-gram, whether the $n$-gram is a true expansion (in some context) for the acronym.

### *5.3.1 Classifier Training Examples*

For *positive* training examples, we used the natural source of the 885 entries on the list of pairings, which happen to be part of the gold-standard set (described in Section 4). In other words, the positive training examples were the entries from human-edited acronym dictionaries in which the acronym and expansion each appeared at least five times in the corpus, and were related by one of the common formation rules learned in Section 3.4.

For machine learning purposes, it was important to have *negative* training examples as well, ideally near misses and about the same number as positive training examples. (We considered performing 1-class classification instead—using only positive examples—but such algorithms generally perform less well than binary classifiers.) Since there was no obvious source for negative examples, we constructed a synthetic set. We paired acronyms in the gold-standard set to $n$-grams that were *not* listed in the gold-standard set as the "correct" expansions.

For example, consider the acronym `bi"x`, which, as shown in Table 3, was paired with 640 possible $n$-grams. Only one, `bit xwlim` (*house-of sick-people*, *"hospital"*), was a "correct" expansion in the gold-standard set, so we designated the pair `bi"x` = `bit xwlim` as a positive training example. To create a negative training example, the acronym was paired with one of its remaining $n$-grams, randomly selected from the remaining list—say, `bin xwwt` (*between farms*, *"between farms"*).

### *5.3.2 Classification Features*

For each pairing of an acronym and $n$-gram, we computed a feature vector containing measures of various properties of the acronym, $n$-gram, and the relationship between them:

- the PMI (pointwise mutual information) of the $n$-gram;
- the collection frequency, document frequency, and inverse document frequency of the acronym;
- the collection frequency, document frequency, and inverse document frequency of the $n$-gram;
- the length of the acronym (in number of letters);
- the length of the $n$-gram (in number of words);
- the formation rule relating the acronym and $n$-gram; and
- the Latent Dirichlet Allocation topic similarity of acronym and $n$-gram [2,3].

All features were straight-forward to compute except the last. Latent Dirichlet Allocation (LDA) is a topic modeling algorithm which discovers hidden (latent) themes in large textual datasets. We used LDA to model topics in the corpus, to capitalize on the intuition that an acronym and its expansion tend to appear in similarly-themed document contexts. For example, if the acronym `bi"x` appears strongly in healthcare-related documents yet weakly in art-related documents, we would expect its expansion, `bit xwlim` (*house-of sick-people*, *"hospital"*) to behave similarly; while a competing expansion, such as `bin xwwt` (*between farms*, *"between*

*farms*"), has a different distribution appearing weakly in both healthcare- and art-related documents.

To implement this observation, we computed features representing the degree of LDA topic similarity between the acronym and its paired $n$-gram. We built an LDA model from the corpus with $T = 300$ topics using the open-source toolkit MALLET [21].We then represented the acronym as a vector $\overrightarrow{a} = (a_1, a_2, ..., a_T)$ over the topic space, where coordinate $a_i$ is the acronym's score for topic $i$ as given by the LDA model.

Similarly, the $n$-gram was represented as a vector $\overrightarrow{e} = (e_1, e_2, ..., e_T)$ where $e_i$ is the $n$-gram's score for topic $i$. Determining the coordinate values for the $e_i$'s were less obvious, as the LDA model provided topic scores for individual tokens, not multi-word $n$-grams. We therefore inferred the $e_i$'s from the topic scores for the *individual tokens* of the $n$-gram in three simple ways:

1. Pointwise multiplication of the individual tokens' scores for topic $i$;
2. Pointwise addition of the individual tokens' scores for topic $i$; and
3. Pointwise addition of the individual tokens' scores for topic $i$, but with a special case ensuring a value of 0 if any of the summands is 0.

For each method of calculating $\overrightarrow{e}$, we then computed the measure of topic similarity between the acronym and the $n$-gram by taking the cosine similarity of the two vectors $\overrightarrow{a}$ and $\overrightarrow{e}$:

$$\text{TopicSimilarity}(\overrightarrow{a}, \overrightarrow{e}) = \frac{\overrightarrow{a} \cdot \overrightarrow{e}}{|\overrightarrow{a}| \cdot |\overrightarrow{e}|}$$

Finally, these three topic similarity measures were included as classification features, representing the degree of LDA topic overlap between the acronym and the $n$-gram.

#### *5.3.3 Classifier Training and Intrinsic Evaluation*

On the 1768 total training examples, half positive and half negative, we trained a support vector machine (SVM) with a linear kernel using the sequential minimal optimization (SMO) algorithm [26]. We also tried several other classification algorithms, including SVMs other than SMO, notably LibSVM [4]; SVMs with nonlinear kernels; and decision trees [27]. However, these other classifiers performed worse. Unfortunately, we did not have enough withheld data to conduct a reliable experiment to choose the most appropriate learning method. All machine learning algorithms were implemented using the open-source Weka suite [13].

For baseline comparisons, we also built two naïve classifiers:

- *Baseline classifier #1:* A simple classifier which selects the highest-frequency $n$-gram paired with the acronym as the expansion and rejects all other $n$-grams for the acronym; and
- *Baseline classifier #2:* A classifier identical to ours (SMO, trained on the same set of training examples), but using only the PMI feature.

For both SVMs, performance was evaluated using 10-fold cross-validation. As shown in Table 4, our classifier easily outperformed the baselines.

| Classifier | P | R | F |
|---|---|---|---|
| Baseline classifier #1 | 0.55 | 0.03 | 0.05 |
| Baseline classifier #2 | 0.61 | 0.59 | 0.60 |
| Our classifier | 0.82 | 0.81 | 0.82 |

**Table 4** Classifier **P**recision, **R**ecall, and **F**-score.

Because our method of constructing negative examples involved an element of chance, we repeated it 10 separate times, training otherwise-identical SMO classifiers on different negative training examples (though with identical positive training examples). We found the standard deviation to be just 0.0083 for precision and 0.0081 for recall; such low numbers indicated high robustness for our method of constructing negative examples.

The classifier provided interesting insights into which features were most influential in its prediction (features that were most prominently weighted). The inverse document frequency and PMI of the $n$-gram were the strongest features (which motivated our designs for the baseline classifiers, making them as strong as possible while remaining simple). The three next-strongest features were the ratios of the inverse document frequencies of the acronym and $n$-gram in two of the sub-corpora and in the total corpus, respectively. Following closely was the LDA topic similarity measure calculated using the pointwise addition method of calculating the $n$-grams' topic scores.

We further explored the impact of the LDA topic similarity features on the classifier's performance. The second (pointwise addition) method of calculating the $n$-gram topic scores $\overrightarrow{e}$ proved most effective, as indicated by the three methods' respective features' relative weights in the SVM. Additionally, Table 5 shows that while holding out the LDA features from the feature set negatively impacted performance by only a few percentage points, training the classifier on *only* the LDA features achieved reasonably good (if still lower) performance, indicating that a great deal of useful information is contained within the LDA topic similarity scores alone.

| Feature Set | P | R | F |
|---|---|---|---|
| All features | 0.82 | 0.81 | 0.82 |
| All features *except* LDA | 0.79 | 0.79 | 0.79 |
| *Only* LDA features | 0.70 | 0.69 | 0.70 |

**Table 5** Importance of LDA similarity features in classifier **P**recision, **R**ecall, and **F**-score.

## 5.4 The Complete Dictionary

The final complete dictionary consists of 11,088 acronyms with expansions from three sources:

1. **Classifier:** All acronym/$n$-gram pairs that the classifier predicted as positive instances.

2. **Gold:** The three manually-compiled acronym–expansion dictionaries used to create the gold-standard set (described in Section 4), including pairs that were not included in that set (those whose acronym and/or expansion did not pass the frequency threshold in the corpus, or which were not related by a common formation rule).
3. **Gematria:** Easily-generated gematria acronyms (explained in Section 3.5), for numbers up to 5800, which covers all Hebrew calendar years up until the current time (more precisely, next century).

Each entry is tagged with meta-data indicating the source, as well as statistical information with respect to the corpus (including LDA topic scores, which proved useful for contextual disambiguation).

## 6 Acronym Disambiguation

As an extrinsic evaluation of the quality of our acronym dictionary we defined the following **acronym disambiguation** task: given an acronym and several possible expansions, determine which expansion is correct for a particular context. The motivation for this task is that a disambiguation method that uses a dictionary will be more accurate when its underlying dictionary is improved.

We randomly selected 202 acronym types out of all acronym types that appeared at least five times in our corpus. For each of them, we identified in the corpus an instance of that acronym along with the sentence and document in which it appeared. These documents, constituting a negligible 0.09% of the total number of corpus documents, were held out of all procedures involved in the dictionary-building process described in Section 5, so as to be eligible for evaluation here.

Native Hebrew-speakers manually analyzed these acronyms within their document contexts and provided the expansions as well as any prefixes or suffixes (as explained in Section 3.3) to identify the "base" acronyms. To ensure high-quality annotation, at least two annotators reviewed each instance, with (rare) disagreements resolved by an additional reviewer.

We used the annotated acronym-expansion pairs as the evaluation set. A randomly-selected subset of 25 (12%) instances were reserved for development, and 10 were discarded as typos or errors, leaving 167 pairs. Of these, 25 were identified by the annotators as gematria acronyms (described in Section 3.5). After the LDA model was trained on the other corpus documents, we inferred LDA topic scores for the held-out documents, as explained in Section 5.3.2.

### 6.1 Baseline Dictionaries

We compared the performance of the dictionary we built with two other dictionaries representing the existing state-of-the-art. We now define the two baseline dictionaries.

Inspired by the most common previous method of acronym dictionary-building (see Section 2), we searched the corpus for Hebrew acronyms that were either immediately followed by a parenthetical clause of at least two words, or were

themselves in parentheses and preceded by 2–4 words—for example, "CIA (Central Intelligence Agency)" or "Central Intelligence Agency (CIA)."

Rather than re-implement existing algorithms that focus on local acronyms only, we decided to manually generate an upper bound for the accuracy of such algorithms. We manually annotated each such case as being a proper acronym/expansion match or not. We were generous in this assessment, even if the non-acronym part was not an *exact* match for the expansion: for example, the sentence fragment "CIA (the government officials at the Central Intelligence Agency)" would have been rated as providing a correct match, even though the parenthetical phrase contained extraneous words beyond the expansion.

This baseline, *baseline dictionary #1*, thus served as an upper bound for the *best possible* acronym dictionary constructed from local parenthetical acronyms. As *baseline dictionary #2* we used the union of the three gold-standard dictionaries of human-curated acronym-expansion pairs from in Section 4.

### 6.2 Dictionary Entry Ranking

For a given acronym, each dictionary typically offered multiple expansion possibilities, to account for different meanings in different contexts. Therefore, the ranking of expansions within dictionary entries had an important influence on performance on the disambiguation task.

First, for each acronym instance to disambiguate, we considered all possible function word prefix analyses if the acronym began with suitable letters (explained in Section 3.3). For example, `btel"d` could conceivably be either a five-letter acronym, or the four-letter acronym `tel"d` prefixed with a `b+` (*in/on*). We considered the analyses in order from shortest to longest prefix; in this case, assuming there was the shortest possible prefix—none at all—and only afterwards guessing that the prefix was `b+` (*in/on*). This decision was based on the observation that shorter prefixes are almost always more likely than longer ones; this inclination was proven beneficial when tested on the development set too.

For the dictionary we built, we ordered by the expansions' sources (described in Section 5.4)—first gematria, then gold, then classifier—as this gave the best results on the development set. Within each source, we ranked entries by the LDA similarity score of the expansion and the acronym's document context. (The calculation was identical to that described in Section 5.3.2, where we computed the LDA similarity score of the acronym and possible expansion $n$-gram. Here, we replaced the acronym topic vector with the inferred document topic vector.)

The entries for baseline dictionary #2 (gold dictionary) had no natural ranking, so we ordered expansions at random within the entry, though again gematria expansions (if any) were always first. Typically there was only one or a few expansions per acronym entry in this dictionary, so the ranking was less important here.

We did not attempt to rank the entries for baseline dictionary #1 (best-possible dictionary of local parenthetical acronyms) because, as we shall soon see, it performed very poorly even under the most generous conditions.

### 6.3 Results

We evaluated the three dictionaries—the one that we built, and the two baselines—with respect to the following test: *Given an acronym and the document it appears in, is its correct expansion (with respect to its context) in the top $r$ results of the dictionary's entry for that acronym?*

We tested four values of $r$: $r = 1$ ("is the very top dictionary expansion correct?"), $r = 2$ and $r = 3$ ("is the correct expansion in the top 2 (or 3) dictionary entries?"), and $r = \infty$ ("is the correct expansion in the dictionary at all for this acronym?"). Performance was measured as the percentage of instances of the evaluation set which passed this test, as shown in Table 6.

| $r$ | **Baseline Dict. #1** | **Baseline Dict. #2** | **Our Dict.** |
|---|---|---|---|
| 1 | | 66.47% | 72.46% |
| 2 | | 77.25% | 79.04% |
| 3 | | 78.44% | 81.44% |
| $\infty$ | 52.38% | 82.63% | 85.03% |

**Table 6** Performance of the three dictionaries on the disambiguation task, given as the percentage of the evaluation set instances which have the correct expansions in the top $r$ results for the dictionary's entries for the acronym.

| $r$ | **ERR vs. Baseline Dict. #1** | **ERR vs. Baseline Dict. #2** |
|---|---|---|
| 1 | | 18% ($p < 0.03$) |
| 2 | | 8% ($p < 0.25$) |
| 3 | | 14% ($p < 0.06$) |
| $\infty$ | 68.56% | 14% ($p < 0.06$) |

**Table 7** Error rate reduction (ERR) of our dictionary, compared to the two baseline dictionaries, on the disambiguation task.

Our dictionary performed well, beating both baseline dictionaries, especially the first (best-possible dictionary of local parenthetical acronyms). Note that because of how we constructed and ranked the entries in our dictionary, it is guaranteed to perform at least as well as the strong baseline dictionary #2; what we are interested in is how *much* better. Since baseline dictionary #2 had high performance as well, looking at the error rate reduction of our dictionary was a better measure of improvement, as shown in Table 7. The $p$ values were calculated using McNemar's paired $\chi^2$ one-tailed test; using the conventional $p < 0.05$ significance level threshold, our dictionary had statistically-significant improvement for the most important $r = 1$ case, and nearly-significant improvement for the $r = 3$ and $r = \infty$ cases.

## 7 Conclusion and Future Work

We described a machine-learning-based method for constructing acronym dictionaries from unstructured texts, including non-local acronyms that are not accompanied in the same documents by their expansions. The resulting dictionaries list acronyms along with their (ranked) potential expansions, and include contextual data that can help in the acronym disambiguation task.

Our work focused on secular Modern Hebrew texts, but our methods are easily adaptable to other languages. Hebrew has a clear advantage regarding the ease of identifying acronyms, due to their specialized orthographic style (as discussed in Section 3.2), and the language is also a good test bed due to the widespread usage of acronyms in texts. However, Hebrew has many special challenges too, including the complexity introduced by its complicated morphology and orthography, particularly prefixed function words (explained in Section 3.3); the general paucity of language processing resources; and common acronym formation rules that involve multiple initial letters from expansion words.

Other languages with complicated morphologies (e.g., Arabic), as well as any resource-scarce language, may especially benefit from our approaches. For languages with non-trivial acronym identification, including English and Arabic, our work would need to be combined with more sophisticated methods of identifying acronyms. However, the other stages of our approach are less language-specific. To adapt them to a new language, the following language-specific resources and considerations are required:

1. A gold-standard set of acronyms and their "correct" expansions, such as an existing hand-built acronym dictionary or acronym entries from a generic dictionary.
2. A large corpus that includes instances of acronyms and their expansions (not necessarily in the same documents).
3. A basic understanding of any language-specific acronym linguistic properties, to which the dictionary-building method would be adjusted. For example, German acronyms may be built from initial syllables instead of initial letters, such as *Stabi* instead of *SB* for *Staatsbibliothek* (*state library*, *"State Library"*), and thus the formation rules approach would need appropriate modification. Similarly, a language not containing an analogy to Hebrew's function word prefixes, such as English, would safely skip all prefix-related considerations (discussed in Section 3.3) made by our described Hebrew method.

Within a language, our work can be applied to specialized genres by simply substituting a genre-specific corpus and gold-standard acronym dictionary. Obvious genres for Hebrew include Israeli military texts and Jewish legal texts, which are both especially rife with acronyms and have good gold-standard dictionaries [14, 1, 20]. Another natural direction is to apply the classifier developed in Section 5—which was trained on general Modern Hebrew—to domains without existing gold-standard dictionaries such as internal corporate documents, specialized research fields, etc.

Furthermore, in this work we used standard classification with rather obvious sets of features. Future extensions of our results can use more elaborate feature sets, including word n-grams and part of speech sequences, and in particular word

embeddings, which have been found extremely useful for a variety of similar disambiguation tasks. The results can very likely be improved through a more careful selection of the features, ablation experiments and fine-tuning of the parameters. We leave such improvements to future research.

While acronyms may be less frequently used in standard English texts than in Hebrew, we note that they are used extensively in social media (in English as well as in many other languages). New acronyms (e.g., *LOL* = *laughing out loud*, *ROTFL* = *rolling on the floor laughing*, *YMMV* = *your miles may vary*) are introduced frequently, and are typically not susceptible to the common local-acronym identification methods of existing approaches. We believe that our dictionary construction approach will be instrumental in addressing these challenges.

## Acknowledgments

The authors are grateful to Ran El-Yaniv, Doug Freud, Assaf Glazer, Shie Mannor, and Shaul Markovitz for their machine learning advice. We thank Rafi Cohen for his help with LDA, Nachum Dershowitz for his historical acronym guidance, Chaim Kutnicki for his efficient coding support, Tomer Ashur and Sela Ferdman for their pre-processing of the Wikipedia corpus, and Josh Wortman for his dictionary assistance. Statistically significant improvements to our math were provided by Nicholas Mader, Breanna Miller, Tony Rieser, Zach Seeskin, and Brandon Willard. Thanks to acronym annotators Yosi Atia, Hannah Fadida, Limor Leibovich, Lior Leibovich, Shachar Maidenbaum, Elisheva Rotman, and Beny Shlevich. This research was supported by THE ISRAEL SCIENCE FOUNDATION (grant No. 1269/07).

## References

1. Ashkenazi, S., Yarden, D.: Treasury of Acronyms. Kiryat Sefer, Jerusalem (1994). In Hebrew
2. Blei, D.M.: Probabilistic topic models. Communications of the ACM **55**(4), 77–84 (2012)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet Allocation. Journal of Machine Learning Research **3**, 993–1022 (2003)
4. Chang, C.C., Lin, C.J.: LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology **2**, 27:1–27:27 (2011)
5. Dannélls, D.: Acronym recognition: Recognizing acronyms in Swedish texts. Master's thesis, Department of Linguistics, University of Gothenburg, Gothenburg, Sweden (2006)
6. Dannélls, D.: Automatic acronym recognition. In: Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics, pp. 167–170. Trento, Italy (2006)
7. Dannélls, D.: Acronym classification using feature combinations (2007)
8. HaCohen-Kerner, Y., Kass, A., Peretz, A.: Baseline methods for automatic disambiguation of abbreviations in Jewish law documents. In: J.L. Vicedo, P. Martnez-Barco, R. Munoz, M.S. Noeda (eds.) Proceedings of the 4th International Conference on Advances in Natural Language, *Lecture Notes in Artificial Intelligence*, vol. 3230, pp. 58–69. Springer-Verlag Berlin Heidelberg (2004)
9. HaCohen-Kerner, Y., Kass, A., Peretz, A.: Abbreviation disambiguation: Experiments with various variants of the one sense per discourse hypothesis. In: E. Kapetanios, V. Sugumaran, M. Spiliopoulou (eds.) Natural Language and Information Systems, *Lecture Notes in Computer Science*, vol. 5039, pp. 27–39. Springer (2008). DOI 10.1007/978-3-540-69858-6_5

10. HaCohen-Kerner, Y., Kass, A., Peretz, A.: Combined one sense disambiguation of abbreviations. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, HLT-Short '08, pp. 61–64. Association for Computational Linguistics, Stroudsburg, PA, USA (2008). URL http://dl.acm.org/citation.cfm?id=1557690.1557707
11. HaCohen-Kerner, Y., Kass, A., Peretz, A.: HAADS: A Hebrew Aramaic abbreviation disambiguation system. Journal of the American Society for Information Science and Technology **61**(9), 1923–1932 (2010)
12. HaCohen-Kerner, Y., Kass, A., Peretz, A.: Initialism disambiguation: Man versus machine. Journal of the American Society for Information Science and Technology **64**(10), 2133–2148 (2013)
13. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: an update. SIGKDD Explorations **11**(1), 10–18 (2009). DOI 10.1145/1656274.1656278. URL http://doi.acm.org/10.1145/1656274.1656278
14. Israel Defense Forces: Dictionary of Abbreviations and Acronyms (2010). In Hebrew
15. Itai, A., Wintner, S.: Language resources for Hebrew. Language Resources and Evaluation **42(1)**, 75–98 (2008)
16. Jain, A., Cucerzan, S., Azzam, S.: Acronym-expansion recognition and ranking on the web. In: Information Reuse and Integration (IRI 2007), pp. 209–214. IEEE (2007)
17. Ji, X., Xu, G., Bailey, J., Li, H.: Mining, ranking, and using acronym patterns. In: Proceedings of the 10th Asia-Pacific Web Conference on Progress in WWW Research and Development, APWeb'08, pp. 371–382. Springer-Verlag, Berlin, Heidelberg (2008). URL http://dl.acm.org/citation.cfm?id=1791734.1791779
18. Li, C., Ji, L., Yan, J.: Acronym disambiguation using word embedding. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pp. 4178–4179 (2015). URL https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9404
19. Mair, C.: Twentieth-Century English: History, Variation and Standardization. Studies in English Language. Cambridge University Press (2009)
20. Marwick, L.: Biblical and Judaic Acronyms. KTAV Publishing House, Brooklyn, NY (1979)
21. McCallum, A.: MALLET: A machine learning for language toolkit (2002). http://mallet.cs.umass.edu
22. Muchnik, M.: Morpho-phonemic characteristics of acronyms in contemporary Hebrew. Hebrew Linguistics **54**, 53–66 (2004). In Hebrew
23. Nadeau, D., Turney, P.D.: A supervised learning approach to acronym identification. In: Proceedings of the 18th Canadian Society Conference on Advances in Artificial Intelligence, AI'05, pp. 319–329. Springer-Verlag, Berlin, Heidelberg (2005). DOI 10.1007/11424918_34. URL http://dx.doi.org/10.1007/11424918_34
24. Okazaki, N., Ananiadou, S., Tsujii, J.: Building a high-quality sense inventory for improved abbreviation disambiguation. Bioinformatics **26**(9), 1246–1253 (2010). DOI 10.1093/bioinformatics/btq129. URL http://dx.doi.org/10.1093/bioinformatics/btq129
25. Park, Y., Byrd, R.J.: Hybrid text mining for finding abbreviations and their definitions. In: Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing, pp. 126–133 (2001)
26. Platt, J.C.: Fast training of support vector machines using sequential minimal optimization. In: B. Schölkopf, C.J.C. Burges, A.J. Smola (eds.) Advances in Kernel Methods - Support Vector Learning. MIT Press (1998). URL http://research.microsoft.com/~jplatt/smo.html
27. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo, CA (1993)
28. Ravid, D.: Internal structure constraints on new-word formation devices in modern Hebrew. Folia Linguistica **24**, 289–348 (1990)
29. Schwartz, A.S., Hearst, M.A.: A simple algorithm for identifying abbreviation definitions in biomedical texts. In: Proceedings of the Pacific Symposium on Biocomputing, pp. 451–462 (2003)
30. Spiegel, Y.S.: The use of uncommon abbreviations and acronyms. Yeshurun (2002). In Hebrew
31. Stevenson, M., Guo, Y., Al Amri, A., Gaizauskas, R.: Disambiguation of biomedical abbreviations. In: Proceedings of the Workshop on Current Trends in Biomedical Natural Language Processing, BioNLP '09, pp. 71–79. Association for Computational Linguistics, Stroudsburg, PA, USA (2009). URL http://dl.acm.org/citation.cfm?id=1572364.1572374

32. Tadmor, U.: The acronym in Israeli Hebrew. Leshoneinu La'Am **39**, 225–257 (1988). In Hebrew
33. Xu, J., Huang, Y.: Using SVM to extract acronyms from text. Soft Computing - A Fusion of Foundations, Methodologies and Applications **11**, 369–373 (2006). DOI 10.1007/s00500-006-0091-5. URL http://dl.acm.org/citation.cfm?id=1180624.1180635
34. Yi, J., Sundaresan, N.: Mining the web for acronyms using the duality of patterns and relations. In: Proceedings of the 2nd International Workshop on Web Information and Data Management, WIDM '99, pp. 48–52. ACM, New York, NY, USA (1999). DOI http://doi.acm.org/10.1145/319759.319782. URL http://doi.acm.org/10.1145/319759.319782
35. Zahariev, M.: Efficient acronym-expansion matching for automatic acronym acquisition. In: Proceedings of the International Conference on Information and Knowledge Engineering, pp. 32–37 (2003)