Coming to Terms: Automatic Formation of Neologisms in Hebrew

Moran Mizrahi *, **Stav Yardeni Seelig** *, **Dafna Shahaf**
The Hebrew University of Jerusalem
{moranmiz,stav.yardeni,dshahaf}@cs.huji.ac.il

Abstract

Spoken languages are ever-changing, with new words entering them all the time. However, coming up with new words (neologisms) today relies exclusively on human creativity. In this paper we propose a system to *automatically* suggest neologisms. We focus on the Hebrew language as a test case due to the unusual regularity of its noun formation. User studies comparing our algorithm to experts and non-experts demonstrate that our algorithm is capable of generating high-quality outputs, as well as enhance human creativity. More broadly, we seek to inspire more computational work around the topic of linguistic creativity, which we believe offers numerous unexplored opportunities.

1 Introduction

Human languages are always changing, evolving, and adapting to the needs of their speakers. New words regularly enter our vocabulary, while others disappear. For example, the word "selfie" (self-portrait digital photo, typically taken with a smartphone) has recently become part of everyday English, even spawning variations such as helfie (a selfie of one's hair), welfie (a selfie taken during a workout), and drelfie (a selfie taken while being drunk) (Christiansen and Chater, 2016).

Newly coined words or expressions are termed *neologisms*. There are many neologism formation mechanisms; common ones include loanwords borrowed from another language (kindergarten), morphological derivation (socialize, simplify), compounding (football, breakwater), blending (spoon + fork = spork), and acronyms (laser).

Importantly, the coining of novel words relies on *human creativity*, with the new terms often conveying a lot of information in an inventive way. In this work, we set out to explore the possibility of **automating some of this inherently-human, creative linguistic process**. In other words, we ask whether computers can generate high-quality, novel words on their own, or alternatively help inspire people to find better words.

We focus on automatic generation of neologisms in the Hebrew language. Hebrew has several properties which make it particularly interesting for our goal: first, modern Hebrew was revived after a long period of time (Rabin, 1963; Fellman, 1973), which is unique. There are no other cases of a natural language without any native speakers subsequently acquiring millions of native speakers. For this reason, foreign words are very common in Hebrew, and many terms need to be coined.

Another reason for focusing on Hebrew is its unusual regularity of noun formation. While portmanteaus (word blends), word combinations and other formation mechanisms do exist in Hebrew, most words are created by combination of root and pattern. To the best of our knowledge, this method of word generation was not explored before in a computational context. Our contributions are:

- We propose a novel task, automating the formation of neologisms in Hebrew, and propose an algorithm mimicking the human process. Our pipeline includes models for learning special-case phonological rules, as well as other statistical properties of the language. We release open-source code and data here.
- We evaluate individual components and then run a user study, comparing our algorithm to both experts and non-experts. While humans are better (as expected), our algorithm is capable of generating high-quality words, winning 27-41% of pairwise comparisons in terms of suitability, likability and creativity, as well as having candidates in the top quartile of the overall ranking.
- In addition to comparing our system to human

* Both authors contributed equally to this paper.

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 4918–4929

performance, we build on ideas from human-computer interaction to explore how the system can *improve* human performance. We show our algorithm's output can enhance human creativity, getting non-experts closer to experts. We believe that this type of evaluation can be beneficial for many NLP tasks, especially creative tasks or tasks where human performance is still significantly superior.

Beyond the specific task of generating Hebrew neologisms, we hope this work would inspire further research towards automating and supporting creative tasks.

2 Background

Hebrew is classified as an Afroasiatic, Semitic language. Like Arabic, Hebrew is written right to left. Vowels are indicated by diacritic marks representing the syllabic onset, or by matres lectionis (consonantal letters used as vowels). Everyday printed Hebrew often omits the diacritic marks, resulting in a highly ambiguous text. For example, **בצל** can be diacritized as "onion", "in a shadow" or "in the shadow" (Shmidman et al., 2020).

Hebrew morphology. Hebrew follows nonconcatenative morphology. It is based on **roots**, consisting of a sequence of consonants (usually three), from which nouns, adjectives and verbs are formed. Thus, different words composed of the same root often have semantically related meanings. For example, the words תִּזְמֹרֶת (tizmoret), זַמָּר (zamar), and זֶמֶר (zemer) all have the root **זמר** (sing), and stand respectively for an orchestra, a singer, and a song.

While in English words are usually formed by adding prefixes and suffixes, in Hebrew the root letters are combined into **patterns**, called mishkalim. The patterns are commonly represented by using the arbitrary placeholder letters **קטל** (k-t-l) for root consonants. Patterns usually include diacritics, vowel letters and sometimes prefixes and suffixes. For example, to form the Hebrew word תִּזְמֹרֶת (orchestra), the placeholder letters **קטל** of the pattern תִּקְטֹלֶת are replaced with the root letters **זמר**.

Even though this concept is simple, there is a significant amount of special cases requiring modifications to the form of the final word. From a sample of the Even-Shoshan dictionary (Even-Shoshan and Azar, 2003), we estimate that $\sim 2/3$ of the roots require some modification. For example, combining the root **רפא** with the pattern תַּקְטֵלָה should have resulted in תַּרְפֵּאָה (tarpe'a). However, since **רפא** is a special root (ends with **א**), it becomes תְּרוּפָה (trufa).

Importantly, many patterns denote specific semantic categories. For example, the pattern קַטָּל (katal) is commonly used to describe professions, as in זַמָּר (singer), טַבָּח (cook), and כַּתָּב (reporter). However, not every category has its matching patterns, and some patterns can denote multiple different categories. For example, the pattern קַטֶּלֶת (katelet) can be used for professions in feminine form, but is also a very common pattern for illnesses.

Formation of Hebrew words. Many world languages have official language regulators, often referred to as language academies (e.g., the Royal Spanish Academy, L'Académie française, the Council for German Orthography). The regulating body for Hebrew is the Academy of the Hebrew Language. One of the Academy's most important roles is creating new words to replace loanwords derived from other languages (Fellman, 1974). The initiative tends to come from the public, seeking Hebrew alternatives for foreign words common in everyday speech. A committee of scholars of language, linguistics, Judaic studies, and Bible discusses the word and suggests a Hebrew replacement. Most new words are built using the root-pattern system (aca, 2020), although compound nouns and portmanteaus (blends) are also used.

We note that even with decades of experience, it is difficult to predict whether the new terms will be picked up by the public. Some words catch on immediately, some take years, and some never do.

3 Methodology

In this section we present our algorithm, ELIEZER BOT-YEHUDA (EBY), named after Eliezer Ben-Yehuda, a lexicographer who was the driving force behind the revival of the Hebrew language in the modern era. We follow the three main ways of forming words used by the Academy of the Hebrew Language: root-pattern, compounds, and portmanteaus. The input to the algorithm is a source word in English, for which we wish to find a Hebrew word. We used English as a mediating language due to the variety of linguistic resources available for it, but the algorithm can work with any other language (see Section 3.3). Figure 1 shows the process for the input word "palette".

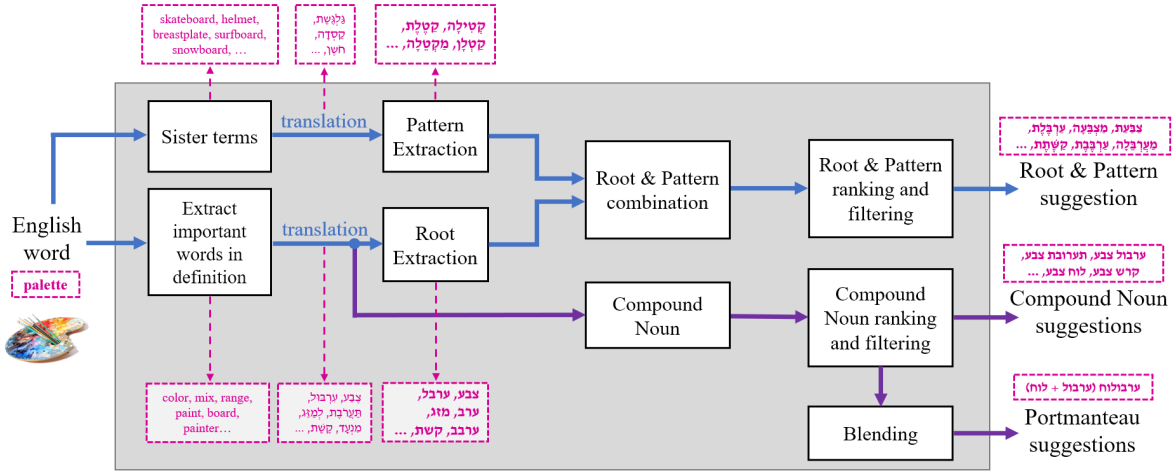

Figure 1: The pipeline of the algorithm, including root-pattern, compounds and portmanteaus, demonstrated on the source word "palette" (see dashed squares). The pipeline mimics the human process of generating neologisms.

3.1 Root and pattern pipeline

Root and pattern combination is the most common mechanism for coining Hebrew terms. We now explain how we simulate this process.

3.1.1 Finding potential roots

The first step towards coming up with a new term is understanding what the word is *about*. Therefore, we created a document for each English word that appeared in our dictionaries, containing multiple English dictionary definitions (from Wiktionary, Merriam-Webster dictionary, WordNet (Miller, 1995), ConceptNet (Speer and Havasi, 2012), Wikipedia abstracts and Easier English Student Dictionary (Rooney and Collin, 2003)). After lemmatizing and removing stop words, we used tf-idf (Ramos et al., 2003) to find the 10 most important words in each document (e.g., color, mix, board for "palette"). Despite the simplicity of this process, it proved to be effective in practice (see section 4.3).

Next, we attempt to identify relevant roots. To do so, we translated the important words into Hebrew, using English Wiktionary, Hebrew Wiktionary, and Hebrew Wordnet (Ordan and Wintner, 2007). Importantly, the output of the translators was diacritized words, from which we extracted roots (identifying the root without diacritics is much harder). Given the translations, we used Hebrew Wiktionary and Even-Shoshan dictionary[1] to identify roots. We ranked the roots based on their important word's tf-idf score. Extracted roots for "palette" include צבע (color), ערבל (mix).

3.1.2 Finding potential patterns

As mentioned in section 2, many of the patterns in Hebrew convey semantic information. Thus, to find patterns reflecting the word's category, we use Wordnet's hypernym and hyponym relations to extract up to k=100 sister-terms of the original foreign word. We translate these into Hebrew, with the hope that some already have Hebrew translations, which could hint at the appropriate patterns.

Hebrew Wiktionary provided roots and patterns for the translated words, but Even-Shoshan dictionary provided roots only; see the end of section 3.1.3 for details on how we inferred the patterns for translations with root only. Finally, we chose the top patterns based on their prevalence. As many semantic categories have several corresponding patterns, and due to sparsity of our resources, we chose to use the top 4 patterns. In the case of "palette", one pattern found was מַקְטֵלָה (maktela), used for instruments.

3.1.3 Combining roots and patterns

A naive combination of a root and a pattern will not necessarily generate the word correctly (section 2). Thus, we trained a seq2seq model to modify the naive root and pattern combination into a valid Hebrew word (תַּרְפֵּאָה → תְּרוּפָה). We did not use a rule-based model due to the large number of rules and to allow a more general pipeline.

We curated a dataset of 3365 words, with root and pattern, extracted from Hebrew Wiktionary. We used the naive combination function on the

[1]Throughout our entire pipeline, we found that Hebrew resources are few and limited, so it was crucial to incorporate different sources to gain coverage. Our repository has pointers to download all free sources.

root and the pattern (substituting root letters in the pattern) to create the model's inputs, and trained it to turn them into the correct Hebrew words. The vocabulary size of the dataset was 46 (including Hebrew letters and diacritics). The dataset was divided into train, validation and test sets with 80%, 10% and 10% of the data respectively.

Model architecture and training details. The architecture is of character-based attentional seq2seq model (Bahdanau et al., 2014) with a single GRU layer. We used a bidirectional encoder with character embeddings and the decoder included dropout. The character embeddings in the encoder were concatenated to binary vectors, indicating for each root letter whether it belongs to different special-case root families (e.g., guttural letters). See Appendix for the choice of model parameters. Example output for this stage for "palette" was מַצבֵּעָה (matsbe'a), a combination of the root "color" (צבע) with the instrument pattern מַקְטֵלָה (maktela).

The model achieved 0.68 accuracy on the test set. Mean Levenshtein edit distance for errors only (after setting the distance of two diacritic characters that sound alike to zero) was 1.63 characters. Most of the differences to ground truth were diacritics differences. For further evaluation see section 4.2.

We also used our model for inferring patterns of dictionary words with root but no pattern in our dictionary. We combined these words' roots with all possible patterns, and let our seq2seq model process them. If the result was identical to the original word, we considered the pattern likely.

3.1.4 Ranking and filtering suggestions

At this stage we had root and pattern suggestions. Next, we wanted to select the more "Hebrew looking" words. This was necessary both since the seq2seq model did not fix all of the possible issues, and since we wanted to make sure the new word suggestions fit into the target language in terms of their statistical characteristics. To choose the best root-pattern combinations per root, we used a character based Hebrew language model. For each combination of root and pattern, the model computed a probability score. We kept the two combinations with the highest probability per root, filtering words with probability ≤ 0.1.

To train our model, we needed a sufficient amount of Hebrew words with diacritics. Therefore, we crawled the Ben Yehuda project website, containing the classics of Hebrew literature [2]. Hebrew is a morphologically rich language. Thus, each token in the text may include multiple morphemes. Since we wanted the language model to represent statistical properties of the words themselves, we cleaned them from prefixes according to grammar rules[3] (see elaboration in the Appendix). The final dataset consisted of 514,300 unique words with diacritics, and 4,955,687 characters, with average word length 9.6 characters. The number of possible characters (including diacritics) was 46. The data was divided into train, validation and test sets (80%, 10% and 10% respectively). We used an n-gram character-based language model. See implementation details and parameter choice in the Appendix. Further evaluation of the model is provided in section 4.3.

To prevent confusion, the last step of the algorithm is to filter out words which are identical or sound like existing Hebrew words (Levenshtein edit distance is zero, with substitution weight of two diacritic characters that sound alike set to zero).

3.2 Compound and portmanteau pipeline

In addition to our main pipeline, we also supported two less-common word formation processes: Compound and portmanteau (see Figure 1). To create proper grammatical compound nouns for a source word, we translate the important words as before (see section 3.1.1). We filter out all important words without a root, to exclude loanwords. Then, we pair up the important words left to create a compound noun, ranking the pairs according to the sum of their tf-idf scores.

To make sure the compound nouns are grammatical, we focus on a specific case of compound noun which is the highly prevalent in Hebrew, and check whether the words in the combination are both nouns and have a "genitive case" relation. This was done using UDPipe POS tagger and dependency parser (Straka and Straková, 2017). An example of a compound for "palette" was לוּחַ צֶבַע (luakh tseva, meaning "color board").

To form portmanteaus, we attempted to blend the top compounds when possible, according to blending rules (Bat-El, 1996). For "palette", one example was עִרְבּוּלוּחַ (irbuluakh, meaning "mix" + "board").

[2] https://benyehuda.org/

[3] https://hebrew-academy.org.il/2013/07/18/אותיות-השימוש-ניקוד/

3.3 A note on generalizability

Even though the scheme we presented focuses on Hebrew, it can be adapted to other languages as well. First, note that the root-pattern system is also used in Arabic (the fifth most spoken language in the world). By changing the data sources and retraining the seq2seq model, our algorithm should also work for this language. In addition, the compound and portmanteaus strategies discussed in the pipeline are common in languages without Hebrew's root-pattern system. Thus, these formation processes can be used in numerous languages.

More broadly, we would like to encourage the utilization of our pipeline and its main components (identifying related-content words, identifying potential word forms, word generation via language-dependent manipulations, ranking outputs using language models) when generalizing the algorithm to other languages. We believe it can serve as a useful guide for automating the creative linguistic process of neologism generation in any language.

4 Evaluation of individual components

Our pipeline (depicted in Figure 1) is composed of several components. In this section we evaluate the contribution of the three main components: important words (tf-idf), combining roots and patterns (seq2seq model) and ranking and filtering (language model). For these evaluations, we used student annotators who are native speakers of Hebrew.

4.1 Important words extraction

For this evaluation, two annotators manually marked words they consider important in 15 English word definitions (20-300 words each). We measured agreement using Jaccard Index, averaged over the words, resulting in 0.4 with std = 0.197. Inspecting the annotations, we note that the annotators tended to mark a relatively small number of important words in each definition.

We took words chosen by both annotators as ground truth, and measured the mean recall, resulting in 0.7 (std = 0.25). As the main purpose of this component is to capture the important words, we consider the results satisfactory.

4.2 Root and pattern combination

A random sample indicated that the seq2seq model applies changes to about 60% of its inputs. Taking a closer look at the results, we noticed that our model was able to learn and correctly apply some Hebrew phonological rules, such as identifying repeating letters and realizing when they should be merged.

It was also able to correctly add and remove diacritics in words (e.g., recognizing that guttural letters cannot get a gemination mark). One of the model's weaknesses was converting diphthongs to monophthongs. Some examples showing the seq2seq model's ability of applying different rules are shown in the Appendix.

To evaluate the model more quantitatively, we asked two annotators to look at 100 word pairs and identify the one that seems to follow Hebrew phonological rules more closely. These word pairs were sampled randomly from words changed by the seq2seq model (by at least one character).

The agreement between annotators using Cohen's Kappa was significant (0.7). Both of the annotators agreed that the modified word was better in 75% of the pairs. They agreed that the modified word was worse only in 10% of the pairs. Therefore, we concluded that the seq2seq model indeed improves the root-pattern combinations.

4.3 Language model score

For the language model evaluation, we used similar methods. First, we qualitatively examined the probabilities assigned by the model to specific words. We found that existing Hebrew words were assigned high probabilities, while words contradicting Hebrew phonological rules, such as those still containing diphthongs, were assigned low probabilities (examples for word probabilities assigned by the language model are shown in the Appendix).

We created 100 groups of words, sharing a root but using 4 different patterns (as described in 3.1.2). We computed our character-LM score for each word, and extracted the highest and lowest scoring words per group. We asked two annotators to label the more "Hebrew looking" word from these word pairs. Cohen's Kappa agreement was again significant at 0.78. Both of the annotators agreed on the higher-rated word being better in 69% of the pairs, and agreed that the higher-rated word was worse in 20% of the pairs. We concluded that the LM indeed manages to capture useful information. As the LM was trained on Hebrew classics, we believe its performance can be improved using more modern data containing diacritics.

5 Evaluating the algorithm's output

After evaluating the main parts of the algorithm, we continue to evaluate its suggestions (including root and pattern, compound and portmanteaus). We address two main questions: (1) How do the words our algorithm generated compare to those generated by humans? (2) Can our algorithm's output boost creativity in humans generating new words?

We note that we do not expect our algorithm to beat human performance. Rather, we set out to test whether it can generate **plausible suggestions**, and whether it can **inspire people** to suggest better words. We considered the following baselines:

1. **Expert suggestions: Hebrew Academy.** The officially chosen Hebrew words, as well as runner-up suggestions discussed by the committee.
2. **Non-expert suggestions:** New word suggestions by human participants (non experts).
3. **Non-expert + EBY.** New word suggestions by non experts, after being exposed to the algorithm's output.

Step 1: Choosing source words. To choose source words for the experiment, we collected recent Hebrew Academy meeting protocols available online [4]. We composed a list of foreign words for which an official Hebrew translation was chosen as well as runner-up suggestions. We found 91 foreign words with at least two suggestions for a Hebrew alternative and translated them to English (our mediating language). We filtered out English words our dictionaries had no translations for, as well as words with a well-known official Hebrew alternative (identified through 3 annotators; words known by at least one person were discarded). We sampled 20 random words from the resulting filtered list.

Step 2: Non-experts. We recruited 4 non-expert student volunteers and showed them the 20 foreign words. For each word, the participants had two minutes to suggest Hebrew alternatives, then they were exposed to the algorithm's output and had one more minute to come up with suggestions. We chose those time constraints after holding trial runs and observing that suggestions slowed down considerably after the first minute.

Our algorithm's output and the non-expert baselines yielded many suggestions. To narrow them down and even the play field, we mimicked the voting process used by the Hebrew Academy when it picks its top suggestions per foreign word: we recruited three more student volunteers, who discussed and agreed on up to top 3 suggestions from our algorithm's outputs and each of the non-expert baseline suggestions independently. The chosen alternatives were then used for the comparison stage.

[4] https://hebrew-academy.org.il/

5.1 Evaluation metrics

The assessment of the new word suggestions is not trivial, and should take into consideration different aspects. We chose to measure **Suitability** (does the new word fit the original meaning?), **Likability** (do you like it?) and **Creativity** (how creative is it?). We believe these three measures provide a comprehensive view of the fit of the words.

We created an online survey and recruited native Hebrew speakers via student mailing lists and groups. Participation was voluntary. In the survey, the participants saw 5 random source words out of the chosen 20. Each source word was followed by 5-10 Hebrew suggestions from all baselines, order randomized. Participants were asked to rate each suggestion with respect to suitability, likability and creativity on a Likert scale of 1-5.

As Likert scale is an ordinal scale, where arithmetic operations should not be conducted, we defined binary versions of our measures. We concluded that the suitability rating must be high (≥ 4) to pass, as the suggestion has to match the original meaning. For likability and creativity, we settled on the more relaxed threshold of ≥ 3. Looking at the distribution of ratings reinforced this decision, as this is also the exact binarization cutoff we would have chosen to get close to 50% positives (see histogram in Appendix). As one could argue for other reasonable thresholds (e.g., 4 for all measures), we report results for them in the Appendix as well.

Finally, we define a combined binary score, **Combined**, capturing whether the user considers the word a good candidate as a whole. To be positive, a user's rating has to pass the three thresholds: 4 for suitability, 3 for likability and creativity.

5.2 Results

The experiment included 177 participants, providing between 20-29 ratings for each suggestion. In this section we analyze the results.

Correlation between the three measures. First, we calculated the correlation between all measures using Spearman coefficient. We found that both

suitability and creativity are positively correlated with likability (0.62 and 0.45 respectively), as expected. The link between suitability and creativity was weaker (0.25), which agrees with our intuition (as many suitable suggestions are not necessarily creative).

Experts vs. non-experts. We now compare baselines 1 (experts) and 2 (non-experts). For each source word, we identified the best suggestion from each baseline (the word with the highest percentage of positive binary ratings). We found that the experts' best alternative surpassed the non-experts best alternative more times in likability and suitability (65% and 55% respectively). However, this was not the case for creativity (45%). For the combined measure, experts won 70% of the time.

These results are compatible with our beliefs that experts perform better than non-experts in general. The Hebrew Academy is an official institute, and thus it might put more emphasis on suitability and likability than on creativity.

Algorithm vs. humans: shared suggestions. Automatically coming up with the same words humans thought of (whether experts or non-experts) is an encouraging sign. When considering human baselines, we used all of their suggestions, before filtering. Our algorithm produced 4 suggestions identical to expert suggestions, and 2 identical to non-expert suggestions. Non-experts generated 7 suggestions identical to experts. When focusing on roots only, for 14 out of our 20 source words, at least one root our algorithm selected also appeared in the expert suggestions (and 16 appeared in the non-expert ones). In comparison, for 17 words, at least one of the non-expert roots appeared in the expert suggestions.

Algorithm vs. humans: How did we fare? To compare the algorithm to the baselines, we ranked the suggestions for all of the source words by the percentage of the positive (**Combined**) votes they received. Table 1 shows the distribution of positions in the ranked list for the different baselines (the bottom line shows the percentage of words from each baseline, unrelated to the ranking). Not surprisingly, the expert suggestions dominate the top quarter, followed by the non-experts. However, our algorithm is still well-represented in the top quarters, despite having fewer candidates in the race. Interestingly, there are more expert suggestions then non-experts in the bottom quarter.

Likert scores are difficult to compare among different people. Thus, we performed one more evaluation. For each person and each source word they saw, we made pairwise comparisons between each two suggestions they ranked, and computed the total percentage of times one baseline beat another. The results are in Figure 2. As these comparisons are computed in the context of the same person, we believe these results reflect user preference. As in the previous evaluation, the human baselines are better than our algorithm, but it does show promise: it wins 35-40% of the time compared to experts, and 27-41% compared to non-experts.

	EBY	**Experts**	**Non-experts**
Top 25%	10.3%	56.4%	33.3%
50-75	18.9%	43.2%	37.8%
25-50	37.1%	17.1%	45.7%
Bottom 25%	52.8%	33.3%	13.9%
Total	29.3%	38.1%	32.7%

Table 1: Distribution of words from each baseline in each quartile, where the words are sorted by the percentage of positive combined (binary) votes. "Total" indicates percentage of suggestions for each baseline. Human baselines are, as expected, winning, but EBY is still well-represented in the top quartiles, despite having fewer total candidates.

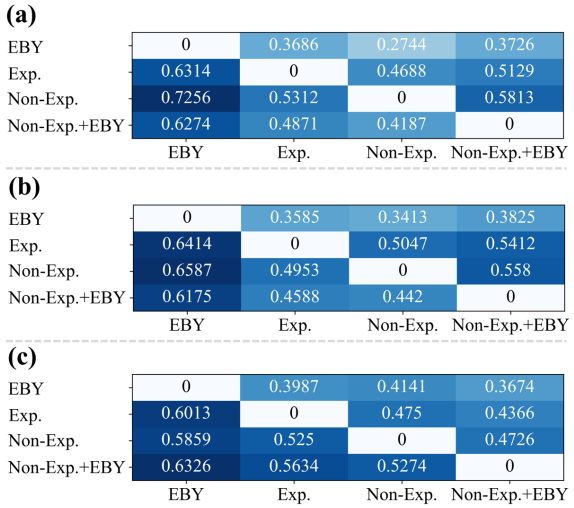

Figure 2: Percentages of times row baseline beat column baseline in **(a)** suitability, **(b)** likability and **(c)** creativity. Comparisons are computed within participant. Showing our algorithm (EBY), experts (Exp), non-experts (Non-Exp), and non-experts added suggestions after seeing the algorithm's outputs (Non-Exp+EBY).

Enhancing human creativity. As noted in the beginning of section 5, we let the non-experts suggest words for two minutes, then showed them EBY's output and collected more suggestions for one minute. We now wish to assess the algorithm's

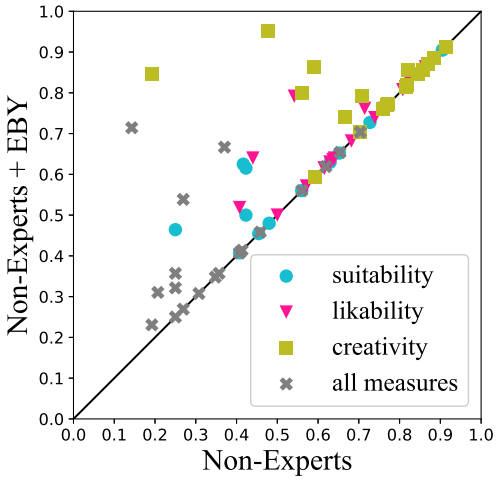

Figure 3: Comparison of the best non-expert suggestion before and after exposure to the algorithm's outputs. X axis is the best non-expert suggestion score before exposure, and y – after. Points above the diagonal indicate improvement.

potential to be a part of people's creative process.

We start by looking at the number of suggestions. The mean number of suggestions before exposure was 11.15 (std = 2.56), and the mean number of additional suggestions after exposure was 8.35 (std = 2.73). The number of additional suggestions is encouraging, as (1) the time after exposure is shorter, and (2) in preliminary trials (without the algorithm's output) we noticed that suggestions were slowing down considerably after the first minute.

After comparing the additional suggestions to the algorithm's outputs, we concluded that they can be attributed to the algorithm in many cases. For example, when translating "guardhouse", participants took a rather rare root suggested by the algorithm (זקפ) and combined it with a better pattern associated with places, resulting in the highest-scoring word in the combined measure: זְקִיפִיָּה (zkifiyah).

Next, we compared the suggestions before and after exposure. Each point in Figure 3 represents a source word. For each suggestion, we compute its score (percentage of positive ratings in the binary measure). The x axis represents the best suggestion's score before exposure, and the y axis – the best non-expert suggestion, either before or after. Words above the diagonal are the ones whose suggestions improved. Exposure to the algorithm improved 20% of the words in suitability and likability. For creativity and the combined measure, 35% of the words improved.

The algorithm's outputs brought the non-experts closer to expert performance. In section 5.2 we compared non-experts to experts. After exposure to the algorithm's outputs, the non-experts' best alternative surpassed the experts' best alternative 45% of the times in the combined measure (compared to 30%), and 70% in creativity (compared to 55%). Three words (זְקִיפִיָּה, חֲלִיפוֹן, סְפָרְפָר) surpassed expert suggestions in all measures. Also refer to Figure 2 to see the effect in terms of pairwise comparisons. Interestingly, the added suggestions beat both the first-round suggestions and the expert suggestions in terms of creativity.

6 Error analysis

We analyzed the algorithm's errors to understand where it is lacking and where to focus future work efforts. We identified two main issues.

Limited resources. In many of the cases in which our algorithm failed to generate appropriate alternatives, it appears to be due to a lack of resources – absent / inaccurate Hebrew translations, or a lack of root / pattern information. For example, consider the word "leggings". One of the important words identified was "fitting", which was inaccurately translated to "appropriate". Another word, "tight", was accurately translated to both הָדוּק (haduk) and מָתוּחַ (matuakh), but our dictionaries did not have their roots. We believe that better Hebrew resources will significantly improve our algorithm.

Connotations. Some of EBY's suggestion received low likability scores. One such word, which was highly disliked, is סָכָל-זֵעָה (sakal ze'a) for "deodorant". Literally, this is a combination of "to thwart" and "sweat". Even though the meaning is well-represented here, both words have a negative connotation. Describing deodorant by the word "sweat" is not appealing, and the Hebrew word for "thwart" also carries negative connotations.

Another example is "periphery", where suggestions focused on roots with meanings of "margin" and "out". This can be offensive for people who live there. In fact, even the Hebrew Academy was unable to reach a decision for this word. After discussing suggestions based on "margin", it was taken off of the agenda following public outrage[5]. We believe a better understanding of connotations can help the algorithm produce more appealing results.

[5]https://tinyurl.com/yd6pq3g7

7 Related work

Lexical creativity. Lexical creativity has been the subject of many studies. Yet, these studies often focus on creative writing of longer texts, such as literature or songs. For example, Settles (2010); Castro and Attarian (2018) focused on developing tools assisting songwriters, and Zhu et al. (2009) predicted human judgments for creativity of sentences. As for lexical creativity work focusing on terms, it mostly explores the cognitive/pyschological aspect of the generation process. For example, Costello (2002) studied the processes guiding word choice when creating noun compounds, and Kuznetsova et al. (2013) explored different contributing factors to creativity in word combinations. In contrast, we explore terms generation from an algorithmic perspective by trying to mimic this process.

Computational neologism. Much previous computational work on neologisms focused on automatic recognition of neologisms and their meanings (Cook and Stevenson, 2010; Cartier, 2017; Costin-Gabriel and Rebedea, 2014; Veale and Butnariu, 2010; Kerremans and Prokić, 2018). Work on computational generation of neologisms mostly focused on creating compounds and word blends from source words (Smith et al., 2014; Deri and Knight, 2015; Gangal et al., 2017; Kulkarni and Wang, 2018; Özbal and Strapparava, 2012; Simon, 2018). Although our algorithm supports these word formations, the main focus of our work is on word generation via root and pattern combination, unexplored in a computational context before. In addition to providing an algorithm for the generation of the neologisms themselves, we also show its potential in enhancing human creativity.

8 Discussion and future work

Coming up with new words (neologisms) is a hallmark of human creativity. In this paper we proposed a system to *automatically* suggest neologisms, using the Hebrew language as a test case. Given a source word, the system identifies related words, roots and patterns and uses them to suggest new terms. We evaluated the system through a user study, comparing it to experts and non-experts, and showed that while humans still perform better, our algorithm is capable of generating high-quality outputs, as well as enhance human creativity.

In the future, we plan to explore more word formation strategies, such as associations; for example, by using the EAT database (Hees et al., 2016). Another exciting avenue is researching the factors influencing the acceptance of new words by the public. A better understanding of successful neologisms, adopted by speakers of the language, can potentially help in their creation.

Beyond the somewhat-niche nature of Hebrew neologisms, we seek more broadly to inspire more work on automating and supporting creative tasks (such as authoring), especially in human-computer collaborative frameworks. We believe more NLP should be applied to tackle psychological phenomena, and that the intersection of the fields opens up many intriguing research questions.

Acknowledgements

We thank the anonymous reviewers for their insightful comments, the Hyadata Lab at HUJI for their thoughtful remarks, the Hebrew Academy for their cooperation, and all the participants in our user studies. We would also like to especially thank Gal Vishne and Raviv Yaniv for their support and help during this project. This work was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant no. 852686, SIAM) and NSF-BSF grant no. 2017741.

References

2020. The Hebrew Academy official website.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Outi Bat-El. 1996. Selecting the best of the worst: the grammar of hebrew blends. *Phonology*, 13(3):283–328.

Emmanuel Cartier. 2017. Neoveille, a web platform for neologism tracking. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 95–98.

Pablo Samuel Castro and Maria Attarian. 2018. Combining learned lyrical structures and vocabulary for improved lyric generation. *arXiv preprint arXiv:1811.04651*.

Morten H Christiansen and Nick Chater. 2016. *Creating language: Integrating evolution, acquisition, and processing*. MIT Press.

Paul Cook and Suzanne Stevenson. 2010. Automatically identifying the source words of lexical blends in english. *Computational Linguistics*, 36(1):129–149.

Fintan Costello. 2002. Investigating creative language: People's choice of words in the production of novel noun-noun compounds. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 24.

C. Costin-Gabriel and T. E. Rebedea. 2014. Archaisms and neologisms identification in texts. In *2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference*, pages 1–6.

Aliya Deri and Kevin Knight. 2015. How to make a frenemy: Multitape fsts for portmanteau generation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 206–210.

Abraham Even-Shoshan and M Azar. 2003. Even shoshan dictionary. *Am Oved, Kineret Zmora Bitan, Dvir and Yediot Aaronot, Tel Aviv*, page 2039.

Jack Fellman. 1973. *The revival of a classical tongue: Eliezer Ben Yehuda and the modern Hebrew language*. 6. Walter de Gruyter.

Jack Fellman. 1974. The academy of the hebrew language: Its history, structure and function. *Linguistics*, 12(120):95–104.

Varun Gangal, Harsh Jhamtani, Graham Neubig, Eduard Hovy, and Eric Nyberg. 2017. Charmanteau: Character embedding models for portmanteau creation. *arXiv preprint arXiv:1707.01176*.

Jörn Hees, Rouven Bauer, Joachim Folz, Damian Borth, and Andreas Dengel. 2016. Edinburgh associative thesaurus as rdf and dbpedia mapping. In *The Semantic Web*, pages 17–20, Cham. Springer International Publishing.

Daphné Kerremans and Jelena Prokić. 2018. Mining the web for new words: Semi-automatic neologism identification with the neocrawler. *Anglia*, 136(2):239–268.

Vivek Kulkarni and William Yang Wang. 2018. Simple models for word formation in english slang. *arXiv preprint arXiv:1804.02596*.

Polina Kuznetsova, Jianfu Chen, and Yejin Choi. 2013. Understanding and quantifying creativity in lexical composition. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1246–1258.

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

Noam Ordan and Shuly Wintner. 2007. Hebrew wordnet: a test case of aligning lexical databases across languages. *International Journal of Translation*, 19(1):39–58.

Gözde Özbal and Carlo Strapparava. 2012. A computational approach to the automation of creative naming. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 703–711. Association for Computational Linguistics.

Chaim Rabin. 1963. The revival of hebrew as a spoken language. *The Journal of Educational Sociology*, 36(8):388–392.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 133–142. Piscataway, NJ.

Kathy Rooney and PH Collin. 2003. *Easier English Student Dictionary: Over 32,000 Terms Clearly Defined, Upper intermediate level.* Bloomsbury Publishing.

Burr Settles. 2010. Computational creativity tools for songwriters. In *Proceedings of the NAACL HLT 2010 Second Workshop on Computational Approaches to Linguistic Creativity*, pages 49–57. Association for Computational Linguistics.

Avi Shmidman, Shaltiel Shmidman, Moshe Koppel, and Yoav Goldberg. 2020. Nakdan: Professional hebrew diacritizer. *arXiv preprint arXiv:2005.03312*.

Jonathan A Simon. 2018. Entendrepreneur: Generating humorous portmanteaus using word-embeddings.

Michael R Smith, Ryan S Hintze, and Dan Ventura. 2014. Nehovah: A neologism creator nomen ipsum. In *ICCC*, pages 173–181.

Robert Speer and Catherine Havasi. 2012. Representing general relational knowledge in conceptnet 5. In *LREC*, pages 3679–3686.

Milan Straka and Jana Straková. 2017. Tokenizing, pos tagging, lemmatizing and parsing ud 2.0 with udpipe. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 88–99, Vancouver, Canada. Association for Computational Linguistics.

Tony Veale and Cristina Butnariu. 2010. Harvesting and understanding on-line neologisms. *Cognitive Perspectives on Word Formation*, 221:399.

Xiaojin Zhu, Zhiting Xu, and Tushar Khot. 2009. How creative is your writing? In *Proceedings of the workshop on computational approaches to linguistic creativity*, pages 87–93.

A Appendices

In these sections we provide more implementation details for the sake of reproducibility, some qualitative evaluations of the models and a short discussion about the choice of our metrics. We release the source, data and train-validation-test splits here.

A.1 Implementation details: Seq2seq

For the seq2seq model described in section 3.1.3, we used AdamOptimizer, with learning rate 5e-4, hidden size 100, batch size 2, teacher forcing ratio 0.65, dropout probability 0.1 and 10 epochs. These hyperparameters were chosen based on accuracy after performing a grid search with the following hyperparameters bounds:

- Learning rate: 1e-4 to 5e-3.
- Hidden size: 10 to 150.
- Batch size: 2 to 16.
- Teacher forcing ratio: 0.5 to 0.8.

10 epochs were chosen based on early stopping.

We also tried other similar models with the same hyperparameter bounds:

- The same architecture, with a unidirectional GRU layer.
- The same architecture without attention.
- No use of character embeddings (one hot vectors instead).
- No use of special case root families information.

The chosen model outperformed all the other options we tried. We trained the seq2seq model on our own laptops, without the use of a GPU.

A.2 Implementation details: Language model

The language model we used in section 3.1.4 is an n-gram character based model, with n=4, and add-k smoothing, where $k = \frac{1}{|V|^4}$ and V is the size of the vocabulary. We normalized the word probabilities according to their length. We chose this model since it had the lowest perplexity (4.72 on the validation set and 4.67 on the test set) compared to other n-gram models with n between 2 and 6 (see Table 2). It also performed better than a one layered GRU language model. In many cases, a language model needs to account for long dependencies between elements (e.g., words). However, this is not the case here, and it is reasonable to assume that the influence of characters within a word is in a small window.

The data for the training of the model was obtained from the Ben Yehuda project website, containing the classics of Hebrew literature. We wanted the language model to represent statistical properties of the words themselves. Thus, we cleaned them from prefixes (מש"ה וכל"ב) using the relevant diacritization rules. The cleaning algorithm used counts of occurrences of words starting with one of the מש"ה וכל"ב letters, before and after removal of their first letter. If the number of occurrences of the word after cleaning was higher than its number of occurrences before that, the letter was removed and the relevant diacritization changes were applied. The prevalence of the definite article ה required a special treatment. To words starting with ה, we applied the changes when the number of occurrences after cleaning was higher than fifth of the occurrences before cleaning. This cleaning procedure was repeated 4 times to account for multiple prefixes (such as in ולכשיצאנו, which should result in יצאנו).

n	Perplexity
2	11.41
3	6.0
4	**4.72**
5	6.37
6	14.64

Table 2: Character based n-gram language model perplexity on the validation set for different n values.

A.3 Qualitative evaluation of the models

When evaluating the seq2seq and language model in sections 4.2 and 4.3, we used both qualitative and quantitative evaluations. We add here some tables demonstrating their qualitative performance.

In Table 3, we show some examples of phonological rules our seq2seq model was able to learn. In Table 4, we show the top and bottom 3 generated Hebrew alternatives for the English word "allergy" according to the probabilities assigned by the language model. This table shows how existing or well formed Hebrew words are assigned with a high probability, while words violating Hebrew phonological rules are assigned with low probabilities.

A.4 Evaluation measures

As Likert scale is an ordinal scale, where arithmetic operations should not be conducted, in section 5.1 we defined a binary score using a cutoff for each of our measures: suitability, likability and creativity.

We chose the cutoffs based on our intuition that suitability must be high (threshold ≥ 4), but lika-

Rules	Input	Output
a	מִשְׁפָּחָה	מִשְׁפָּחָה
b	חִבְבָה	חִבָּה
c	זָוֵב	זָב
d	מֵנְפָה	מַפָּה
e	הִתְלַהֲבוּת	הִתְלַהֲבוּת

Table 3: Examples showing the seq2seq model's ability of applying different rules. (a) lenition (b) uniting repeating latters under a gemination mark (c) diphthong to monophthong (d) assimilation followed by gemination (e) diacritization changes due to guttural letters.

Rank	Word	Probability
1	רֶגֶשׁ	0.44
2	חֲמִירָה	0.40
3	צְפִיָּה	0.39
30	מֶשֶׁ	0.04
31	מִשְׁוֹן	0.03
32	גִּובוֹן	0.01

Table 4: Examples for word probabilities assigned by the language model. We present the top and bottom 3 new Hebrew alternatives for the word "allergy", after sorting all of the outputs according to the language model probabilities. It is evident that the top words are well formed, sometimes already existing, Hebrew words, while the bottom words do not fit to the statistical characteristics of Hebrew words.

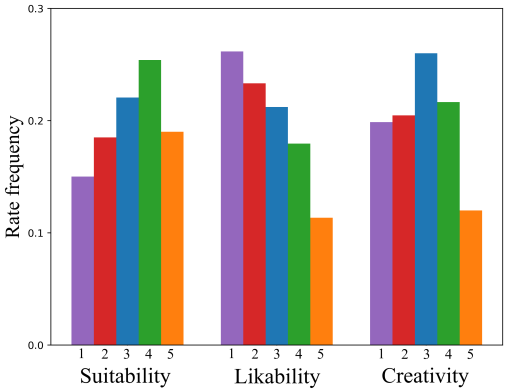

Figure 4: Histogram of ratings for each measure in the user study.

bility and creativity can be more relaxed (threshold of $\geq$ 3). Looking at the distribution of ratings reinforced this decision, as this is also the exact binarization cutoff we would have chosen to get close to 50% positives. See histogram of ratings in Figure 4: for suitability, roughly 50% of the

	EBY	Experts	Non-experts
Top 25%	7.89%	57.89%	34.21%
50-75	30.56%	33.33%	36.11%
25-50	32.43%	32.43%	35.14%
Bottom 25%	47.22%	27.78%	25%
Total	29.3%	38.1%	32.7%

Table 5: Distribution of words from each baseline in each quarter, where the words are sorted by the percentage of positive combined (binary) votes as in Table 1 of the paper, with binarization cutoff 4 for all three measures.

	EBY	Experts	Non-experts
Top 25%	5.4%	59.46%	35.14%
50-75	25.64%	35.9%	38.46%
25-50	40%	25.71%	34.29%
Bottom 25%	47.22%	30.56%	22.22%
Total	29.3%	38.1%	32.7%

Table 6: Distribution of words from each baseline in each quarter, where the words are sorted by the percentage of positive combined (binary) votes as in Table 1 of the paper, with binarization cutoff 3 for all three measures.

participants exceed the $\geq$ 4 threshold. However, for likability and creativity to be close to 50% we needed to treat 3 as a positive label as well.

As one could argue for other reasonable thresholds, we report these results here as well. Tables 5 and 6 are computed the same way as Table 1 in the paper. For Table 5 we use $\geq$ 4 threshold for all measures; in Table 6 we use $\geq$ 3 threshold for all measures. While the top quartile results are lower, the qualitative effect is the same, and the algorithm still has many suggestions in top quarters.