

# “Set up son? Scam set, asserts Bob”: Semi-Automatic Generation of Bilingual Palindromes

Anonymous Authors 

address  
website

---

## Abstract

---

We analyse the problem of composing bilingual palindromes, for which the only high-quality example known is more than 150 years old. We formalise the problem and introduce multiple partial solutions for computer-assisted palindrome composition, as well as a comparison between the composition difficulties in 15 different pairs of languages.

**2012 ACM Subject Classification** Information systems → Structure and multilingual text search; Theory of computation → Algorithm design techniques

**Keywords and phrases** Palindrome, Computational linguistics, Recreational mathematics

**Funding** *Anonymous Authors*: funding

## 1 Introduction

*Evoles ut ira breve nefas sit; regna!*

This sentence is iconic as a linguistic anomaly, as it is the only known example of its kind. Taken in reverse, it gives “*Anger? ’Tis safe never. Bar it! Use love.*” Despite some liberties taken with the translation and typographical symbols, it is the only existing bilingual palindrome with the same meaning in both languages. It was discovered — or at least, first published — by James C.P in an issue of *Our Young Folks Magazine* [2], a children’s magazine. Despite being then — and remaining so ever since its publication more than 150 years ago — our only example of such a palindrome being possible, it was presented with no fanfare by the magazine at all, in small characters on the last page, among other letters to the editor (under a letter featuring one of the simplest palindromes: “Madam, I’m Adam”), with the last name of its creator not even indicated.

The Latin phrase roughly translates to “go forth, in order that anger might be a shallow wrong (over which) you prevail”. While the palindrome’s English version is clearly not a perfect replication, it does communicate the sense of the Latin appropriately. James C.P was the first to point out that when read forwards, the palindrome is not exactly classical Latin, but it is grammatical. The English version is both grammatically cohesive and in line with the Latin in tense and aspect.

Until recently, a legend permeated the anglophone world, which treated this example as the only bilingual palindrome in existence, barring some examples made up of one or two words. This is, unsurprisingly, untrue, although examples are still exceedingly rare. Even when removing the constraint of having a similar meaning in both languages, only two sources appear to mention such bilingual palindromes. The first is a 16-page internal publication linked to the Oulipo and the Collège de ’Pataphysique, written by Luc Étienne in 1984 [8]. The second is a book published by Gérard Durand on 20/02/2002 with a print run of a few hundred copies<sup>1</sup>, which features two pages on the subject [5]. The palindromes

---

<sup>1</sup> The exact number is unknown but inferred from the single copy present as legal deposit.

in both works are apparently not ones where the meaning is similar in both languages<sup>2</sup>. Moreover, they are not perusable online, and the only available online source that mentions a new bilingual palindrome features one that is not fully grammatical<sup>3</sup> [3].

This anomaly — and the associated legend (which the authors initially believed) — is linked to a diversity of reasons. A probable explanation is that Latin and English form a pair that is uniquely suited to such constructions. However, the number of fluent Latin locutors — especially the ones with the expertise and leisure to tackle such problems — has declined greatly since the 19th century.

We then propose to use computational tools to help modern palindrome composers create new bilingual palindromes between arbitrary pairs of languages (as long as they share a segmental script).

## Contributions

Our contributions are threefold:

- We formalise some of the concepts associated with bilingual palindromes, which give rise to a few non-trivial algorithmic problems. We also introduce several algorithms to generate word lists that can be used as primers for manual composition of bilingual palindromes (which were used to create the palindrome in the title).
- We apply those methods and compare the relative difficulties of bilingual palindrome composition between the six following languages: English, French, German, Polish, Spanish, and Swahili.
- We put a final nail in the coffin of the legend that James C.P.’s palindrome is the only one of its kind (although it remains to this date the only bilingual palindrome with matching meanings).

## 2 Preliminaries

We will consider pairs of languages  $(\mathcal{L}, \mathcal{M})$ , and the reverse languages  $\mathcal{M}^R$  and  $\mathcal{L}^R$ , consisting of the words of  $\mathcal{L}$  and  $\mathcal{M}$  with reversed letter order.

As the languages can feature slightly different diacritics or character sets, we generally follow the convention of removing them (as well as any typographical symbol) to simplify the composition of palindromes, so we will concentrate on a set of characters corresponding to the 26-character Latin letters present in English.

### 2.1 Types of bilingual palindromes

We can define three main kinds of bilingual palindromes.

*Bilingual strings*, which are parsable in both  $\mathcal{L}$  and  $\mathcal{M}^R$  with at least one decomposition in each language. For example, *wakazaa kazirika* in Swahili (a prefix–verb phrase meaning “then they gave birth”, and the root form of the verb “to be angry”) reversed corresponds to *a kir i zak a az a kaw* in Polish (“and pall and student and until and coffee”), and although both can be parsed, they do not have meaning.

---

<sup>2</sup> We say apparently, as the existence of Étienne’s work was not known to the authors until very recently, and the rarity of the surviving copies of this internal publication makes them hard to consult.

<sup>3</sup> The featured palindrome is taken from the Oulipo, and reads as follows: “*Ted, I beg, am I not ever a venom?*” which nearly works in French as “*Mon Eva rêve ton image, bidet!*”

*Bilingual sentences*, which are not only parsable in both  $\mathcal{L}$  and  $\mathcal{M}^R$  but follow a semblance of grammatical rules in both. The rules can be bent somewhat, as is frequently done already in palindrome composition and related word games. For example, **Sir, o, blame Diana** reversed corresponds to **An aide mal Boris** (where **An** is a first name) in French [5].

*Bilingual meaningful sentences*, which are not just grammatical but make sense in both language even with limited context. A possible example is **I am near, I repel as a rat.** which becomes **Tara, sale, périra en mai.** There is a sub-type of this bilingual palindrome, which we could call a “true bilingual palindrome” where the meanings of both sentences are related, but the only palindrome corresponding to this is the one featured at the start of the introduction.

## 2.2 Objectives

There is one central objective, which is to generate bilingual palindromes of the third kind, ideally ones with related meanings. However, short of being able to do that directly, we can look at simplifying the constraints by characterising good sub-languages from which humans — who excel at pattern seeking — can hopefully work more efficiently to manually compose palindromes.

One possibility would be to generate many potential bilingual strings and choosing the best ones. This is because generating bilingual strings is theoretically easy: they correspond to words from  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ . We take a standard representation of  $\mathcal{L}$  as a prefix tree. We can then obtain a non-deterministic automaton corresponding to  $\mathcal{L}^*$  by adding  $\varepsilon$ -transitions from each terminal node to the root. The intersection  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$  can then be naturally computed by checking the product automaton. The issue with that is one of size: many languages used here have more than 300 000 word forms (including plurals and declensions), and the corresponding prefix trees can have more than a million nodes. Lazy evaluation can help, but this can still be prohibitively computationally expensive.

Besides the computational cost, however, is the fact the overwhelming majority of these strings will be nonsensical, with the probability increasing with the string length — even by uniformly drawing only three words from an English dictionary, one already at most a 5% chance of getting a grammatically correct sentence<sup>4</sup>. A second option is then to find a fixed point in  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ , or in  $\mathcal{L}_s^* \cap (\mathcal{M}_s^R)^*$  for sub-languages  $L_s \subseteq L$  and  $M_s \subseteq M$ . This is the focus of this article and the objective of the algorithms shown in the next section.

One point that we must be careful with is that we have two opposite constraints. If we look at extremely reduced  $L_s$  and  $M_s$ , our work becomes easier in that we get more flexible words, but we might end up with sets too reduced to compose any complex palindrome. On the other hand, if  $\mathcal{L}_s^* \cap (\mathcal{M}_s^R)^*$  has tens of thousands of words, we are too weakly constrained to aid our search efficiently. For example, the following list of 23 words correspond to the palindromic words present in both French and English. That is, for  $\mathcal{L}$  = French, and  $\mathcal{M}$  = English, the list is  $\mathcal{L} \cap \mathcal{L}^R \cap \mathcal{M} \cap \mathcal{L}^R$ :

```
{a, ana, bob, eme, ere, gag, kayak, non, pep, pop, radar, reifier, rotor,
sagas, selles, sexes , shahs, sis, solos, sus, tallat, tot, tut}
```

One can arrange the words in the list in any way one wants and have bilingual strings, but they are too constrained to make long, meaningful sentences. Thus, we are looking for a sweet spot to help human creation.

---

<sup>4</sup> This estimate was done empirically by drawing many sentences, but corresponds to what one would expect from the proportion of the different grammatical classes in English.

## 2.3 Language and dictionary choices

For this study, we chose a first set of six languages, from different — although sometimes related — language groups: English, German, French, Spanish, Polish, and Swahili. As the dictionaries chosen matter a lot for the end results, here are the details of the choices we made.

- **English (278 794 words):** the Collins Scrabble Words word list (formerly SOWPODS) was combined with the three one-letter English words **i**, **a**, and **o**, and with words of length  $> 15$  from Moby Words II (collected via [6]).
- **French (323 422 words):** the French word list from the Hunspell wordchecker was used, collected from [10]. Characters with diacritics were replaced by the equivalent non-modified characters (i.e. **é** → **e**).
- **German (675 659 words):** a word list designed for wordgames was collected from [7]. Special characters were replaced as follows:

ä, ö, ü ß all other characters	æ, oe, ue ss equivalent non-modified character
--------------------------------------	--

- **Spanish (635 039 words):** a word list was collected from [12]. Special characters were replaced by the equivalent non-modified characters.
- **Polish (3 087 991 words):** a Polish Scrabble word list (collected from [11]) was combined with the six one-letter Polish words (**a**, **i**, **o**, **u**, **w**, **z**) and with words of length  $> 15$  from the Hunspell Polish word list (collected from [10]). We had to make some tough choices for special characters and followed phonetic rules, replacing as follows:

ż w all other characters	w v <sup>5</sup> equivalent non-modified character
--------------------------------	--

- **Swahili (67 966 words):** the Hunspell Tanzanian and Kenyan Swahili word lists were collected from [10] and combined. Special characters, all consisting of punctuation, were removed.

## 3 Algorithms

### 3.1 Precomputation

To make the algorithms more efficient, the first step is to remove words that are definitely not in  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ . We can then compute the **n**-grams present in  $\mathcal{L}$ , that is, subwords of length **n** present inside words of  $\mathcal{L}$ . For a given word to be in  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ , for each **n**-gram in the word, the **n**-gram has to be either an **n**-gram of  $\mathcal{M}^R$ , or split between multiple words of  $\mathcal{M}^R$ . For example, VENT is a 4-gram of the French word ECRIVENT, but does not exist as a 4-gram in reversed English. However, it is present in the reverse of IT NEVER.

If we denote the **n**-grams of  $\mathcal{L}$  by  $\mathbf{n}(\mathcal{L})$ , a first idea is then to compute  $\mathbf{n}(\mathcal{L}) \setminus \mathbf{n}((\mathcal{M}^R)^*)$ . The issue with this method is that, even if we compute  $\mathbf{n}((\mathcal{M}^R)^*)$  using dynamic programming, the set has a size exponential in **n**, which generally becomes unwieldy around **n** = 7.

---

<sup>5</sup> This was done to maintain distinguishability from **ż**, and did not introduce further confusion as **v** is not in the Polish alphabet.

We do this for  $n < 7$  and remove words from  $\mathcal{L}$  that have  $n$ -grams not in  $(\mathcal{M}^R)^*$ , getting us a new language  $\mathcal{L}_2 \subseteq \mathcal{L}$ . We can then go the other way and remove from  $\mathcal{M}^R$  words that have  $n$ -grams that are not present in  $\mathcal{L}_2^*$ . We can proceed iteratively until reaching a fixed point. This fixed point is generally reached in 2 to 5 iterations, depending on  $n$ , with some examples shown as Figure 1 and Figure 2 in Section 4.

### 3.2 General algorithms

We can aim to compute three different intersections:

- the words present in  $\mathcal{L} \cap (\mathcal{M}^R)$ , such as `snores`, which exists in both English and reversed French.
- the words present in  $(\mathcal{L}^* \cap (\mathcal{M}^R)) \cup (\mathcal{L} \cap (\mathcal{M}^R)^*)$ , such as `set on` which exists as it is in English, and as `notes` in French.
- the words present in decompositions of strings from  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ , which can include arbitrary long sequences of words before a common decomposition can be found.

The first kind can be computed intuitively by checking the list of words in  $(\mathcal{M})^R$  against a prefix tree for  $\mathcal{L}$ . That said, this list is generally too small to be directly useful, as is shown on Figure 4.

To compute the second kind, one can operate in the same way, except that the prefix tree becomes a non-deterministic automaton, with  $\varepsilon$ -transitions from final nodes to the root. This allows a word to have multiple decompositions. Although we were expecting this set to be too reduced to be useful, it is in fact already nearly too large to work with in practice for certain language pairs, as shown on Figure 5.

As the previous set is already too large, the third set becomes a priori unnecessary. It is, however, the set whose computation introduces non-trivial algorithmic questions. As we said above, the intersection language itself can be computed by getting the product of the automata made from prefix trees for  $\mathcal{L}^*$  and  $((\mathcal{M})^R)^*$ . The issue is that this automaton does not allow us to trivially eliminate bad candidate words (that cannot be in a bilingual palindrome). There are sufficient conditions: for example, if any node reachable by reading this word and going back to the root (on at least one dimension) can reach an accepting state, the word will appear in a decomposition. But this is not a necessary condition.

We can then create dependency graphs to simplify the process. Our goal is then to make sure to link each word to the words in the other language that are necessary for it to be in the fixed point, and look at cycles within this new graph.

For every word  $x \in \mathcal{L}$ , we start by splitting it into two halves (for every possible split into potentially empty subwords):  $A$  and  $B$ . Then we know that  $A$  must be the ending of a word in  $\mathcal{M}^R$ . That is, there must be  $y \in \mathcal{M}^R$  such that  $y = CA$ , with a word  $z \in \mathcal{L} = DC$ , and so on until both words align. Similarly,  $B$  must correspond to a node in the prefix automaton of  $\mathcal{M}^R$ , either reaching a final node or being splittable into two subwords (reflecting the process of  $A$ ).

We then get two options. First we could try to compute those sequences of words until we reach either a synchronising prefix and suffix or an empty set. The latter would show that the  $x$  cannot be in the fixed point of  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ . The former would give us a string with at least one decomposition in each language, with all the words from those decompositions being in the fixed point. The issue with this is that the direct way to do it is to compute — potentially lazily — a product automaton. To see whether  $A$  is a compatible prefix, we need the product of  $(\mathcal{L}^R)^*$  and  $(\mathcal{M})^*$ , except that the latter is rooted in any potential node

corresponding to  $A^R$ . To see whether  $B$  is a compatible suffix, we need the product of  $\mathcal{L}^*$  and  $(\mathcal{M}^R)^*$ , with the former being rooted in any node reachable by  $B$ .

The issue with this method is that we once again run into complexity issues, as the product automaton still has an unwieldy size. We can then weaken our constraint temporarily to partially resolve this.

To do this, we construct four prefix automata, for  $\mathcal{L}^*$ ,  $(\mathcal{L}^R)^*$ ,  $\mathcal{M}^*$ , and  $(\mathcal{M}^R)^*$ . For every final node  $x$  in  $\mathcal{L}^*$ , we look at potential splittings into two halves  $A$  and  $B$ . For each splitting, we follow  $A^R$  in  $\mathcal{M}^*$ , and compute a set of accepting sets (each set is composed of the last node reached, plus any final node whose  $\varepsilon$ -transition we went through). We do the same with  $B$  by following  $B$  in  $(\mathcal{M}^R)^*$  and computing the accepting sets. We then take the set of set products.

By computing the union of all those sets, we get a set of accepting sets, each of which is sufficient to ensure that  $x$  is in  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ . If the set of accepting sets is empty, however, the word cannot be in  $\mathcal{L}^* \cap (\mathcal{M}^R)^*$ .

Using all those accepting sets, we add dependency relations, which are represented by directed hyperedges from nodes in  $\mathcal{M}^*$  and  $(\mathcal{M}^R)^*$  to final nodes in  $\mathcal{L}^*$ . We do the same for the final nodes in  $(\mathcal{M}^R)^*$ .

We then get a quadripartite directed hypergraph. For every node that has no hyperedge pointing towards it, we do the following:

- delete all the hyperedges the node is a part of as a final node;
- if the node has descendants, remove its *final* indicator and the  $\varepsilon$ -transition to the root ;
- if the node has no descendants, remove it entirely, and remove ascendants recursively until reaching one that is final or has another descendant;
- do the same operations for the corresponding node in the reversed tree of the same language.

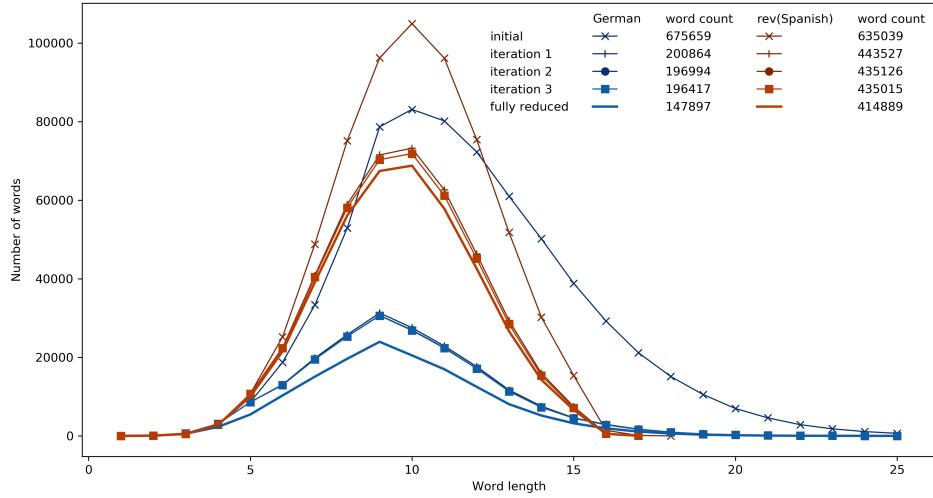
We repeat this process until the set of nodes with no incoming hyperedge becomes empty.

Alas, this method ignores the synchronisation mentioned above, but could already eliminate many words in certain contexts. Instead of computing the full product automaton, a hybrid solution would be to follow this and for each incomplete prefix and suffix to compute the first few levels of this product. For example, one could do a breadth-first search in parallel in both  $\mathcal{M}^*$  rooted in  $A^R$  and in  $(\mathcal{L}^R)^*$ , to try to find either a path leading in parallel to final nodes, or to show that there exists no such path.

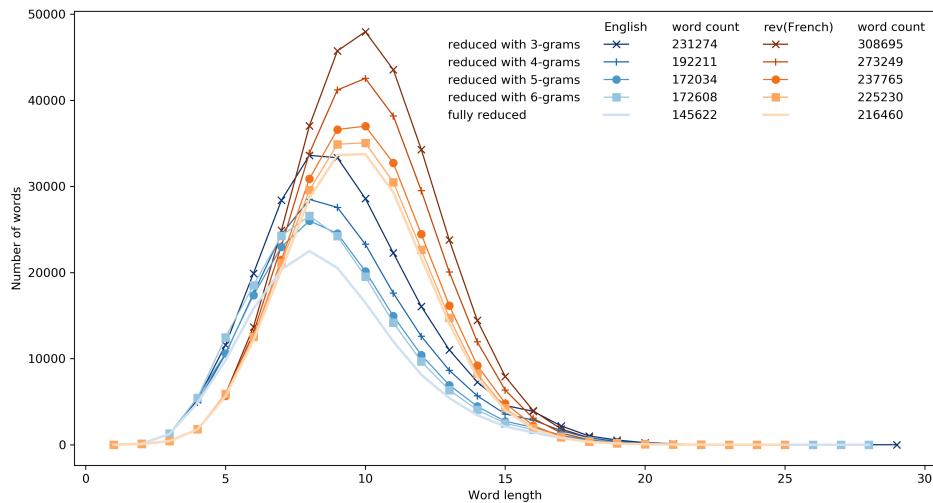
## 4 Data analysis

This section shows the result of the first few methods mentioned previously, to give an idea of what happens in practice on the languages considered.

We first focus on the percentage of words that gets deleted from each dictionary when we apply the **n**-gram elimination method until convergence for a pair of language. This depends on **n**, but it is easy to see that larger **n** remove a superset of words when we restrict ourselves to words of length at least **n**.



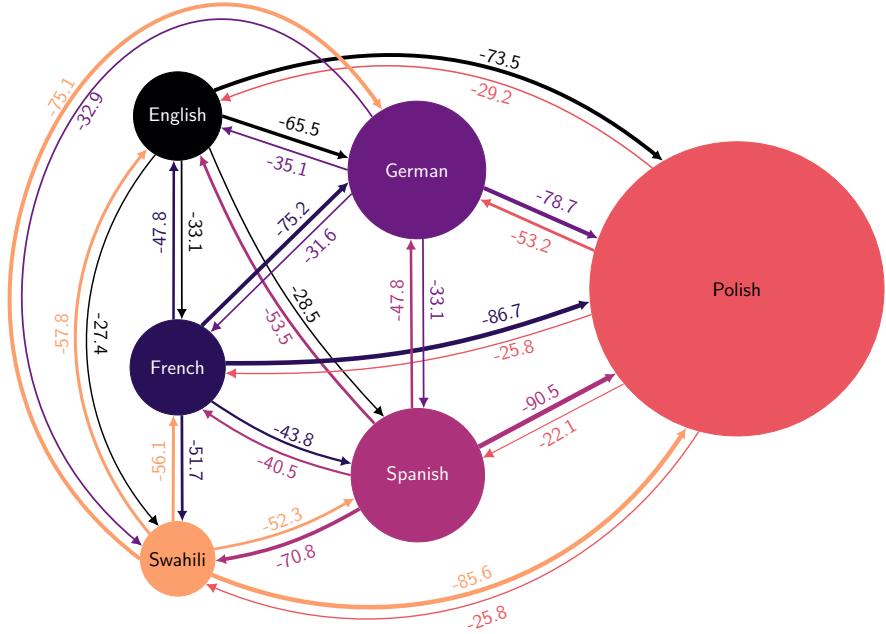
**Figure 1** Effect of removing the 6-grams using the German–Spanish pair, with curves corresponding to the different iterations. The fully reduced curve corresponds to removing the full set (including the n-grams for  $n < 6$ ).



**Figure 2** Effect of removing the n-grams using the English–French pair until convergence, for different values of n, as well as the intersection of all computed sets.

We can see that 6-grams are more efficient at eliminating words, as they correspond to stricter constraints. We can also see that there is generally a last step in the iteration that removes words with very rare n-grams that depended on other rare words, with the last iteration sometimes only removing a few dozen words (Figure 1 and Appendix).

Figure 3 shows that the susceptibility of a language to vocabulary elimination depends on size, orthographic variation, and typological relationship to the other languages. Polish has an extremely high average elimination proportion: this is both due to its size and the high frequency of rare bigrams such as sz. We can observe several other patterns, such as Swahili’s tendency to eliminate vocabulary from other languages and English’s relatively low elimination proportion in all its language pairs, but they could be statistical artefacts.

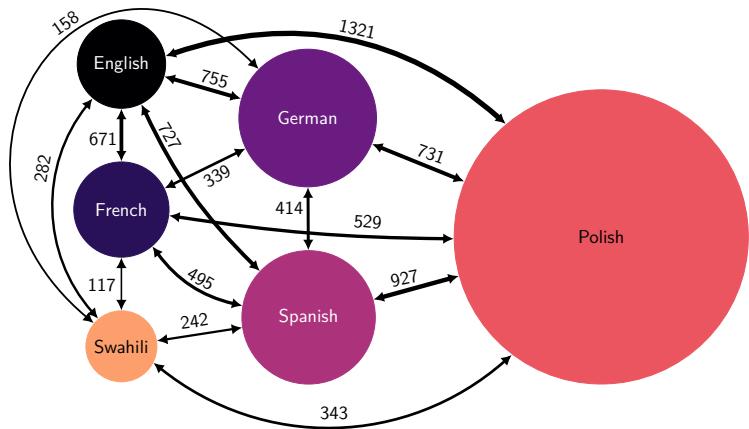


**Figure 3** Graph representing the proportion of words eliminated in each language pair by using the n-gram procedure until convergence. The arrow from  $\mathcal{L}$  to  $\mathcal{M}$  represents the proportion of words eliminated from  $\mathcal{M}$ . For example, Swahili removes 85.6% of Polish words, but Polish removes only 25.8% of Swahili words. The node radius grows with the square root of the language size, and the edge thickness with the proportion of words eliminated.

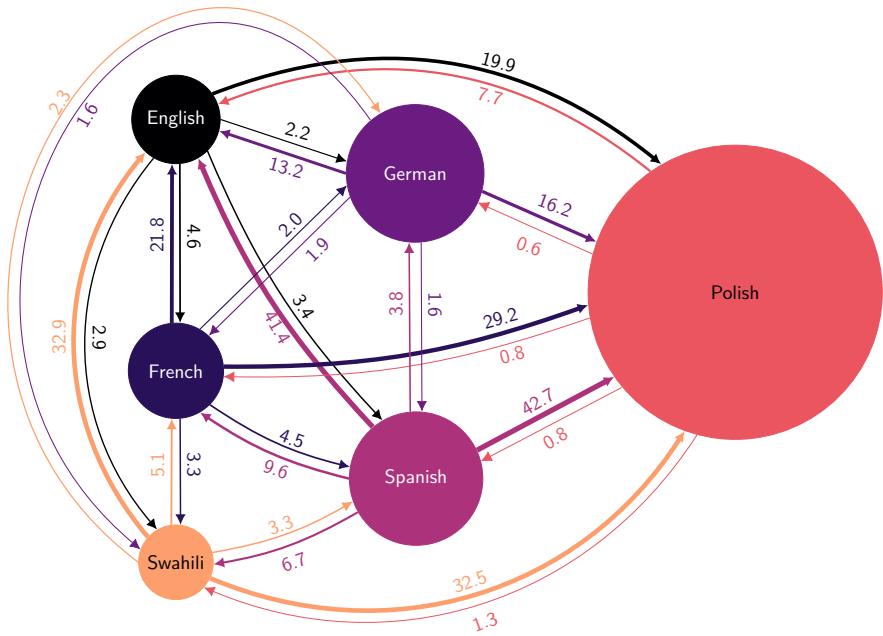
Instead of eliminating as many words as possible, we can also go from the other end to find words in the fixed point. Figure 4 shows the strictest fixed point, corresponding to the number of full words that are shared between  $\mathcal{L}$  and  $\mathcal{M}^R$ , and Figure 5 shows the number of words that can be fully decomposed from one language to a reversed other.

Looking at Figure 4, we can see that the number of common words depends is related to the dictionary sizes. While the size of the Polish dictionary does inform the strong overlaps it has with all the other dictionaries, this factor does not explain the fact that it has nearly three times as many words in common with English as it does with French, which are comparably sized. This, rather, is probably due to the high variability of English orthography on account of its tendency to borrow word formation patterns from other languages [9].

Comparing Figure 3 with Figure 4 shows a potential inverse correlation between common-word overlap and n-gram elimination. For instance, English has the largest overlap with Polish and eliminates the least of its vocabulary. This is not easily generalisable, as we can also see that while Spanish has twice as many words in common with Swahili as does French, it eliminates almost 20% more of its vocabulary. Conversely, between Figures 4 and 5, the size of the common word list does not inform the mutual decomposability: French and Swahili share the fewest words, but French is highly decomposable into Swahili.



**Figure 4** Graph representing the number of bilingual palindrome words between pairs of languages. The node radius grows with the square root of the language size, and the edge thickness with the square root of the size of the number of bilingual palindrome words.



**Figure 5** Graph representing the proportion of words left in each language pair by only allowing perfect decomposition of one word into many. The arrow from  $\mathcal{L}$  to  $\mathcal{M}$  represents the proportion of words from  $\mathcal{L}$  that can be decomposed into words from  $\mathcal{M}^R$ . For example, 16.2% of German words can be parsed in reversed Polish, whereas only 0.6% of Polish words can be parsed in reverse German. The node radius grows with the square root of the language size, and the edge thickness with the square root of proportion of words decomposed.

## 5 Making palindromes: the importance of grammar

Sadly, the efforts shown in the previous section do not lead to language sets of reasonable sizes that are both constrained enough to be manageable and large enough to be useful. One central reason why the word sets that are left are of limited help is that, with this restricted vocabulary, one has to handle grammatical constraints and still make sense.

Returning to the original palindrome, it seems that the key to its unlikely existence is within the language itself. Latin allows a leeway that many of the modern Romance languages don't — its word order is relatively free, allowing subject, object, and verb to come in any order, though this does effect meaning[4].

The freedom this creates is particularly useful in building palindromes. For instance, if one wanted to create a bilingual French-English palindrome, one would have to take into account the strict subject-verb-object order of both languages. Thus, a phrase containing a transitive verb would require the word(s) forming the verb to be in the middle and those forming subject and object to be one another's reversible counterparts. The word(s) in question must be part of the narrow strata of each language's lexicon which will reverse into a word in the other language, while making functional sense as both subject and object. In theory, then, Latin or other languages without a fixed word order — or with a flexible order, such as with Polish — make for easier target languages in the creation of bilingual palindromes.

“Evoles ut” is able to sidestep this somewhat by its arguable lack, in the English translation, of a separately expressed subject to some of the verbs. For instance: “bar” as used here is a two-place predicate, requiring both an object and subject to fulfil it and thus make it grammatical. In English this usually looks like SVO, “[you] bar it”. However, the “you” here is implied by the use of an imperative aspect, so the subject is already built into the verb — a separate expression of it is unnecessary. Thus there is more space for wordplay, as there are fewer building blocks required for the palindrome to be grammatical in English. Essentially, the fewer additional characters required to fulfil a predicate and the more flexibility with word order, the easier it will be (in theory) to manually build a bilingual palindrome.

In this regard, Latin and Polish are both ideal. The issue with the former is that it is becoming hard to find people proficient enough to accomplish the last step of mental gymnastics to create meaningful palindromes. The issue with the latter is its heavy reliance on bigrams that are rare in other languages, whether reversed or not. For example, 51% of the words in our Polish dictionary feature a ‘z’, and 8.5% of them feature an ‘sz’, whereas the corresponding values for English are respectively 4.2% and 0.002% (and 0.003% of ‘zs’).

Short of using those languages, we should then rely on decorated dictionaries with indications of grammatical roles potentially held by words. However, there are few complete dictionaries featuring such information in languages other than English, and palindromists often bend some grammatical rules to achieve their ends. This practice and the fact that rare words are very frequently used in palindromes also complicate data-based methods (such as looking online whether a specific sequence of words has ever been used).

Until those problems are solved, making true bilingual palindromes will then keep a mystical aspect<sup>6</sup>.

---

<sup>6</sup> Indeed, the palindrome's function is not neutral, and early palindromes often communicated a moral. They also had a mystical aspect and were used to inscribe verses of the Torah and to aid in rituals [1].

---

**References**

---

- 1 Ernest Alfred Wallis Budge. *Amulets and talismans*. New Hyde Park, NY: University Books, 1930.
- 2 James C. P. Letter to the editors. *Our Young Folks*, 2(3), 1866.
- 3 Laurence De Looze. *The letter and the cosmos: how the alphabet has shaped the Western view of the world*. University of Toronto Press, 2016.
- 4 Andrew M. Devine and Laurence D. Stephens. *Latin Word Order. Structured Meaning and Information*. Oxford University Press, 2006.
- 5 Gérard Durand. *Palindromes en folie*. Les Dossiers d'Aquitaine, 2002.
- 6 Dwyl. english-words, 2019. URL: <https://github.com/dwyl/english-words>.
- 7 enz. German wordlist for tanglet and other wordgames., 2019. URL: <https://github.com/enz/german-wordlist>.
- 8 Luc Étienne. *Palindromes Bilingues*. Cymbalum Pataphysicum, 1984.
- 9 Suzanne Romaine. Variability in word formation patterns and productivity in the history of english. In *6th International Conference on Historical Linguistics*, pages 451–465, 1985.
- 10 titoBouzout. Hunspell utf8 dictionaries, 2019. URL: <https://github.com/titoBouzout/Dictionaries/>.
- 11 Turekj. Msc project implementation of manisero & turekj, 2014. URL: <https://github.com/turekj/msc>.
- 12 zeke. an-array-of-spanish-words, 2019. URL: <https://github.com/words/an-array-of-spanish-words>.

## 6 Appendix: n-gram curves for all pairs of languages

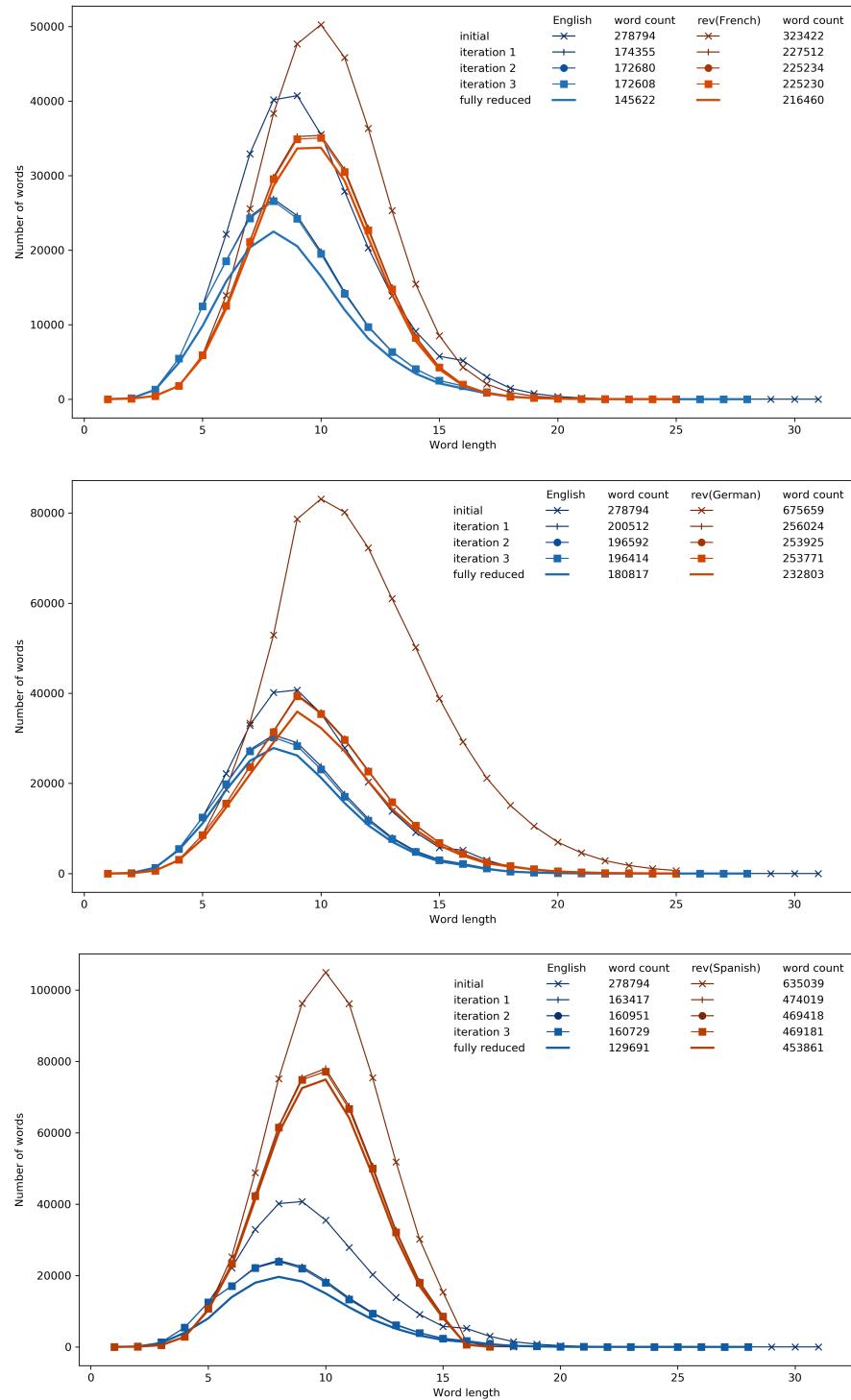


Figure 6 6-gram curves for English-French, English-German, and English-Spanish.

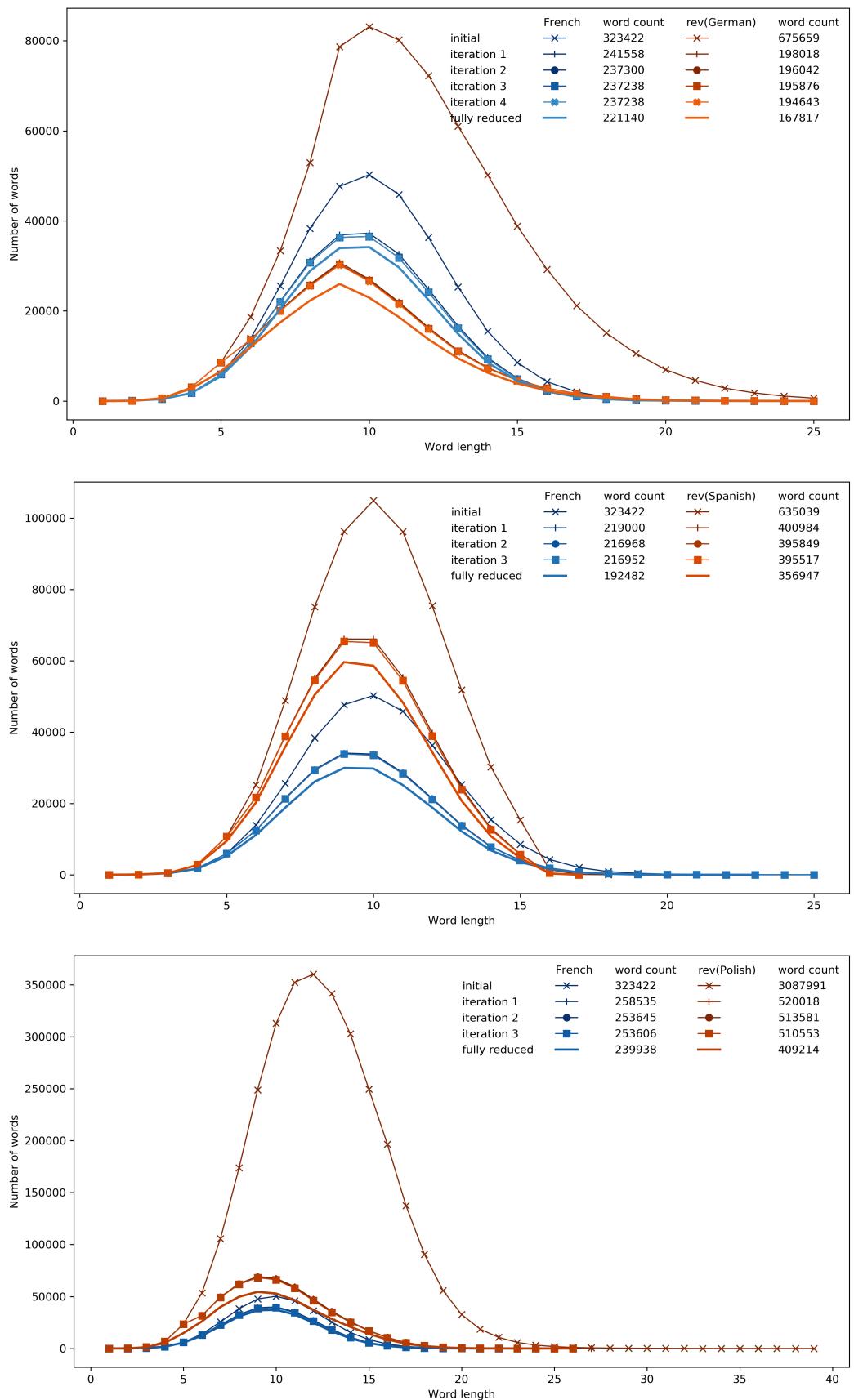


Figure 7 6-gram curves for French–German, French–Spanish, and French–Polish.

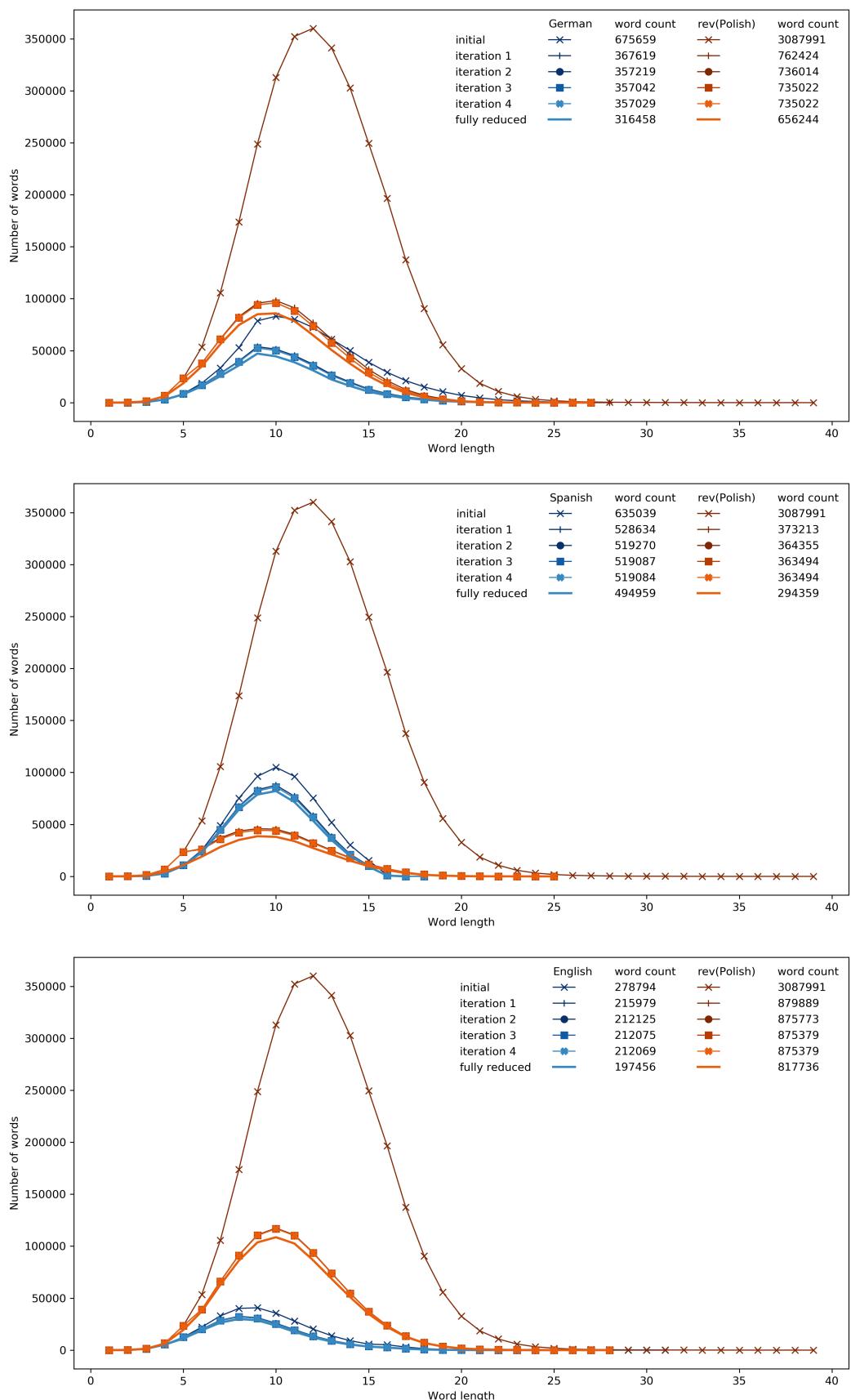
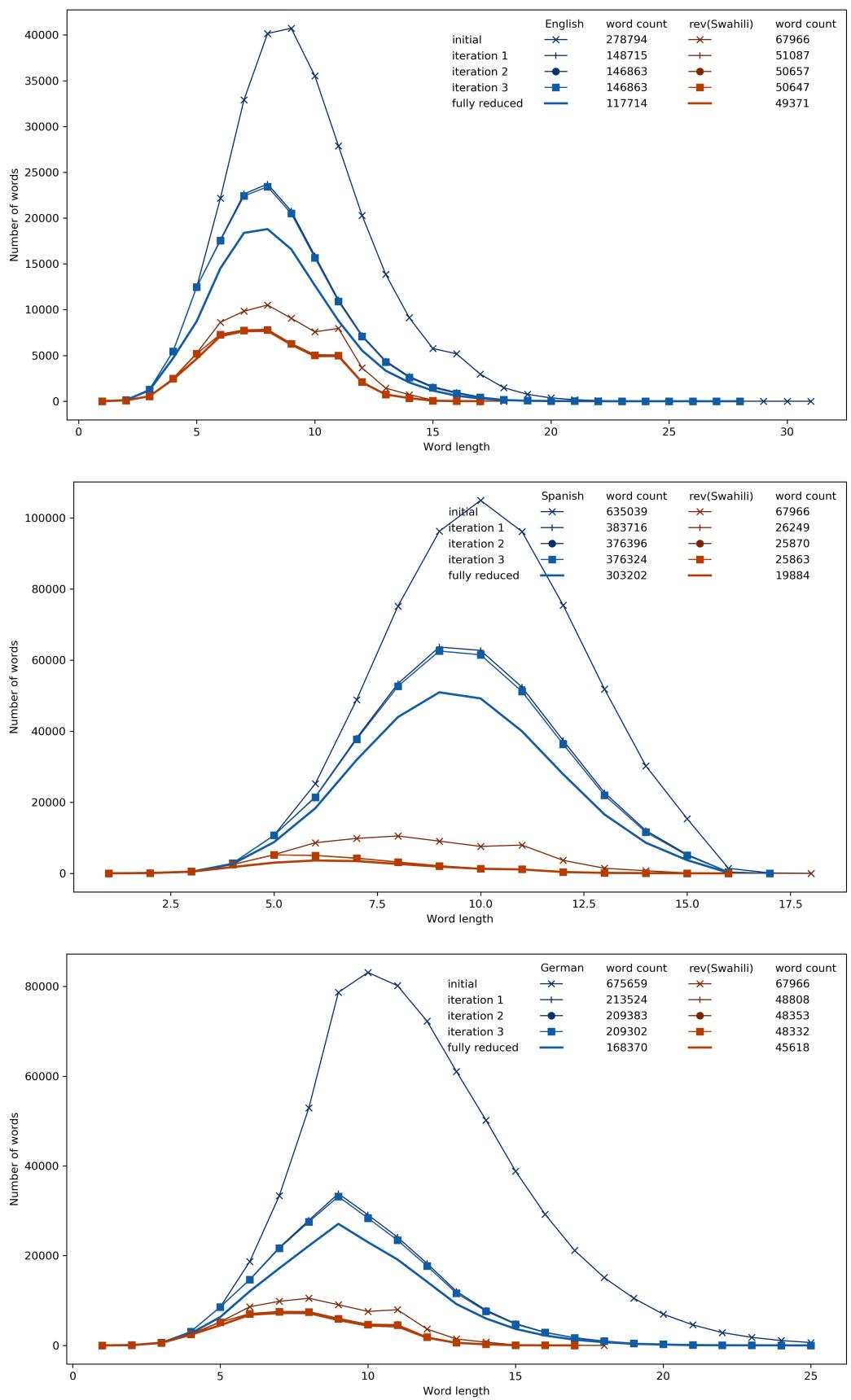
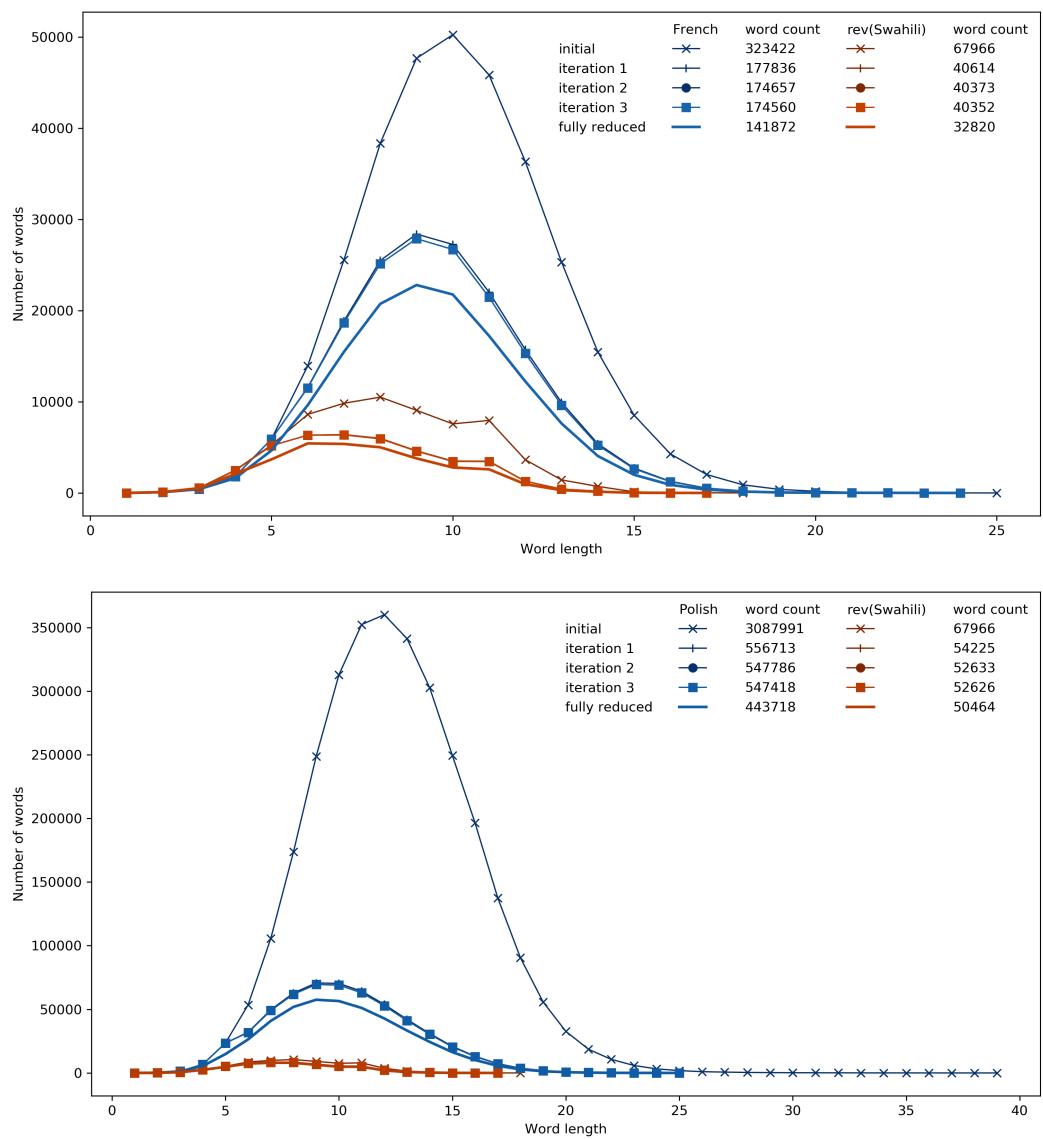


Figure 8 6-gram curves for German–Polish, Spanish–Polish, and English–Polish.



■ **Figure 9** 6-gram curves for English–Swahili, Spanish–Swahili, and German–Swahili.



**Figure 10** 6-gram curves for French–Swahili and Polish–Swahili.