AI Approaches Overcome Variability Problems in Diachronic Text Analysis: The Case of Identifying Bound Affixes in Middle English

Hagen Peukert[1]

[1]Universität Hamburg, ZFDM, Germany

Abstract

This contribution pursues two objectives. First, a short summary of the computational approaches for identifying Middle English affixes comprising different AI methods will be devised reaching from stand-alone application [1] and (semi-automated) shared-work solutions [2][1] to requesting the OED RESTful API.[2] Second, the role of AI and how it is entertained in the context of diachronic affix identification will be shown.

Keywords

Computational Diachronic Linguistics, Middle English, Lexical Affixation, Morphological Language Change

1. Introduction

Collecting representational quantitative data on the frequency of lexical affixes throughout 700 hundred years of English language use has proven to be challenging. While type frequencies of all suffixes and prefixes were determined with relative ease, the identification of token frequencies from larger text corpora took more than a decade of intensive and cumbersome methodological work. Extracting all representations of one affix type and its exact quantities required taking into account all kinds of variability in form and usage. Yet exact quantities are needed to make the more interesting statements on affix productivity and identification as well as interrelations with other factors of influence in the system of language, i.e. a correlation to word order or predictions of likely future changes. Because of the small quantities of available text training material, automated AI approaches have long been ignored as possible candidates for a viable solution. As the following description of the genesis of different computational approaches will reveal, a mixture of an intelligent algorithm and automated matching from existing resources and moving towards micro services generates best results that are useful for progress in linguistic research of diachronic morphology.

[1]https://morphilo.readthedocs.io/en/latest/index.html

[2]https://gitlab.rrz.uni-hamburg.de/softwaretools/morphochron

Humanities-Centered AI (CHAI), Workshop at the 45th German Conference on Artificial Intelligence, September 19, 2022, Trier, Germany

hagen.peukert@uni-hamburg.de (H. Peukert)

0000-0002-3228-316X (H. Peukert)

2. Stages of development and algorithmic refinement

The methodological procedure improved in bumps and leaps over the year as both the linguistic knowledge of Middle English (ME) was enhancing (very moderately) and the inventory of computational methods increased (rather exuberantly) over the years. Hence the phenomenon under investigation as well as the methods helping to understand the phenomenon are constantly changing over time. The researcher has to keep track of all advancements on either aspect, integrate technological innovations in ongoing methodological work, but also consider new findings in the field of derivational morphology altogether. It needs little justification to see that the longer a research project takes, the more problematic it will be to constantly adjust the methodological set up. Yet, besides increasing one's personal learning, such challenge bears the advantage of pervading the subject to an utmost degree and by that diminishing the chance of encountering inconsiderate evidence. It is under this lens that roughly four phases of different approaches to the collection of data can be observed since the inception of the project about one decade ago.

The first phase is characterized by the basic idea of having the machine support manual data collection in text corpora. It derived from elaborated corpus searches for selected derivational morphemes. Realizing that the variations in writing would not produce correct quantities, it soon became clear that a more elaborated algorithm needs to be developed. What was needed foremost was a comprehensive list of all known affixes and possibly all of their known variations. Already at the time, the OED online version (4.0) provided a seemingly complete collection of about 800 suffixes and 300 prefixes. Although this early study of English affix usage could not manage to reliably find all tokens that are really required for generating productivity measures over time, at least the existence of each of the morphemes, i.e. type frequencies, could be proven in textual data in predefined time spans [3]. And it initiated some more intensive investments on designing more sophisticated software programs shifting the methodological work towards artificial intelligence and semi-automated natural language processing.

The second phase started with the production of the Morphilo Toolset [4, 1]. It consisted of three modules (MorExtractor, Morphilizer, MorQuery) that could best be described as independent software components loosely connected by a relational database or other forms of persistent storage. Whereas MorQuery was meant only as a tool to filter and search the final results from the database with more comfort than by command line, MorExtractor and Morphilizer comprise somewhat thoughtful language processing and automated workflows. MorExtractor has two goals. First it manages the text corpora, checks formats and prerequisites such as tagging schemes or time intervals. Second, it matches the available enumerations of suffixes and prefixes to the collections of tagged or plain text corpora respectively considering the word class and time dimension. As an output the matched words together with the corpus data and the word class are saved in a text file, which is the input of the next module. The Morphilizer-algorithm is geared to how a human would analyze a given multi-morphemic word into stems, roots, and affixes, but also the order and position in case several affixes are stringed together, e.g. *tempt-ation-al.* The rules for analyzing are kept simple:

1. possible prefix candidates are matched one by one from the word beginning until no item in the prefix enumeration finds a match (left to right)

2. possible suffix candidates are matched by the same manner from the end of the word (right to left)
3. the remainder, the part that could not be matched, is kept in memory as a potential root of the word
4. the process is repeated in reverse order, i.e. starting with the suffixes
5. if, at the end of the both matching processes, the the two potential roots are equal and the length is greater one the process starts with the next word; otherwise the process starts with the next prefix variant

Furthermore, to avoid the embeddedness problem, the length of an affix chain is taken as a simple criterion to prioritize possible matching candidates, i.e. *-ation* beats *-tion* in e.g. *temptation* leaving as the root *tempt*. The simple logic behind this decision results from probability measures and the relation between the length of a word and its frequency of occurrence. And it follows from Zipf's law.

More precisely, the workflow goes as follows. The file containing lexically annotated words is created by an automaton, which extracts all nouns, verbs, and adjectives from tagged corpora and marks the grammatical morpheme, if regular (plural, 3rd person sing, possessive, progressive/gerund, past, and participles), otherwise it is already tagged in the standard PENN annotation. Enumerated lists serve as a data structure for efficiently keeping suffix and prefix morphemes and their allomorphic representations. There are more than 800 suffix variations; the prefixes comprise about one third of the suffixes.

These affixes were copied from the OED (version 4.0). At the time of devising the architecture, the time of usage, inception, and possibly attrition of the affix was not provided (other than in later versions) and, hence, could not be incorporated into the data type. So the algorithm could be refined with knowledge of the word class, but not on the time of usage, which is only then included in automatic matching if existent in the data base, i.e. it was added by manual post processing. The enumerated lists are processed according to the above algorithm.

Post-processing is done after the words of a text were synchronized with existing data in the master data base. All non-matches are displayed as a result of the algorithm. Corrections can be done by click and drop. Confirming a correct lexical segmentation means that they are saved to the data base together with the number of occurrence in the corpus and the time span of the text.

The gist of phase three is a community based approach [2], in which the programmed Morphilo components would be integrated into a multi-user and web-based platform. It soon became apparent that even a semi-automatic approach done by a single researcher would still exceed any sensible workload. Thus the idea came up to engage other researchers in the field. In analogy to the wiki(pedia)-approach, in which a resource is built step by step by a community profiting from each contribution made. However, the decisive difference for third generation Morphilo is that access to the database is only granted if a contribution is actually made, that is, a user inputs its text and needs to review all the words of the text that are not yet available in the database. In addition, to control quality of the reviews, a certain number (say 10) has to hand in the same lexical segmentation of a word before it is stored in the database. Figure 1 reveals the architecture of the community based approach. One can still identify the three components of the previous version depicted by screenshots.

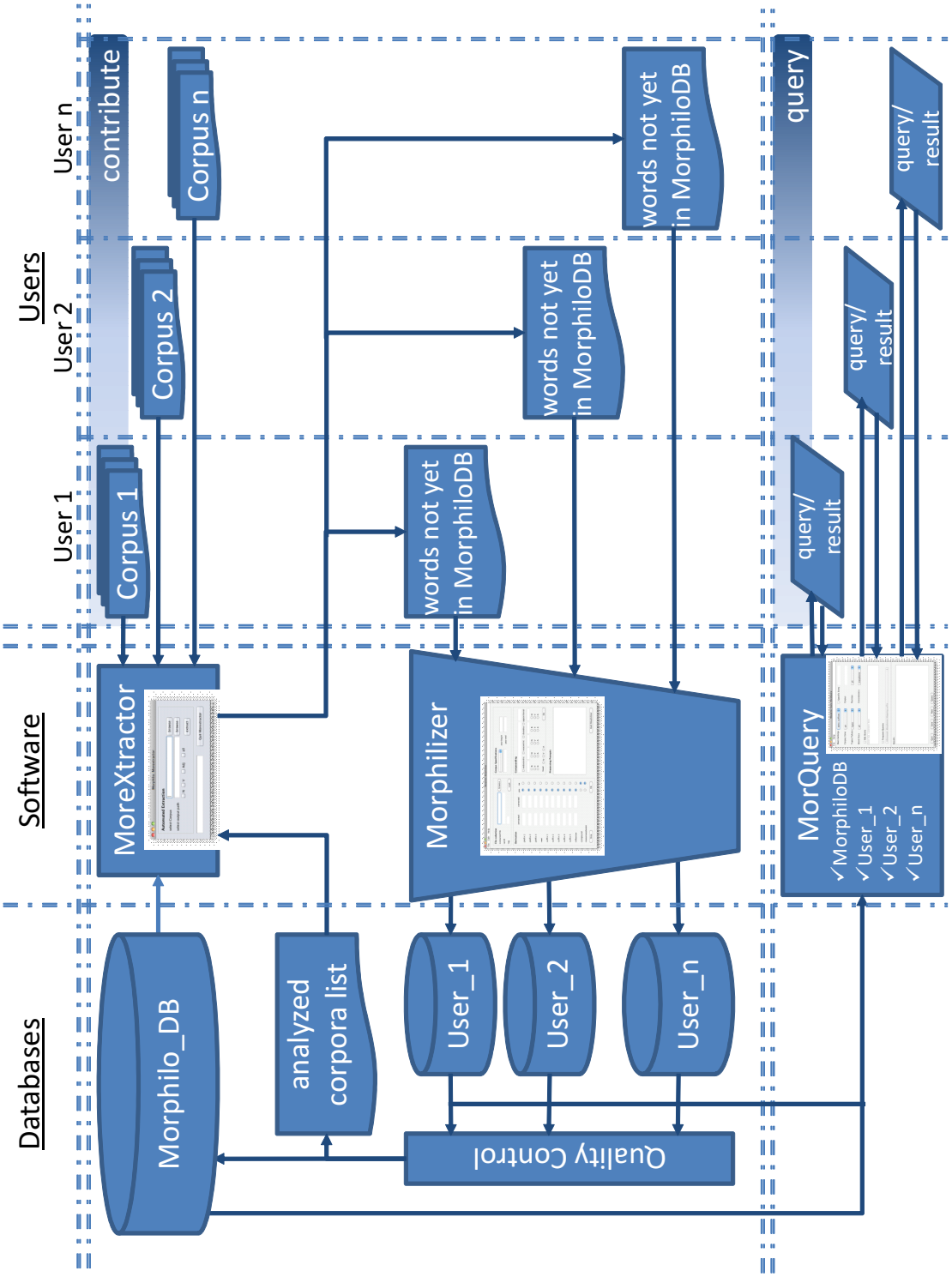

Figure 1: Architecture of the Take-and-Share Approach (3rd generation)

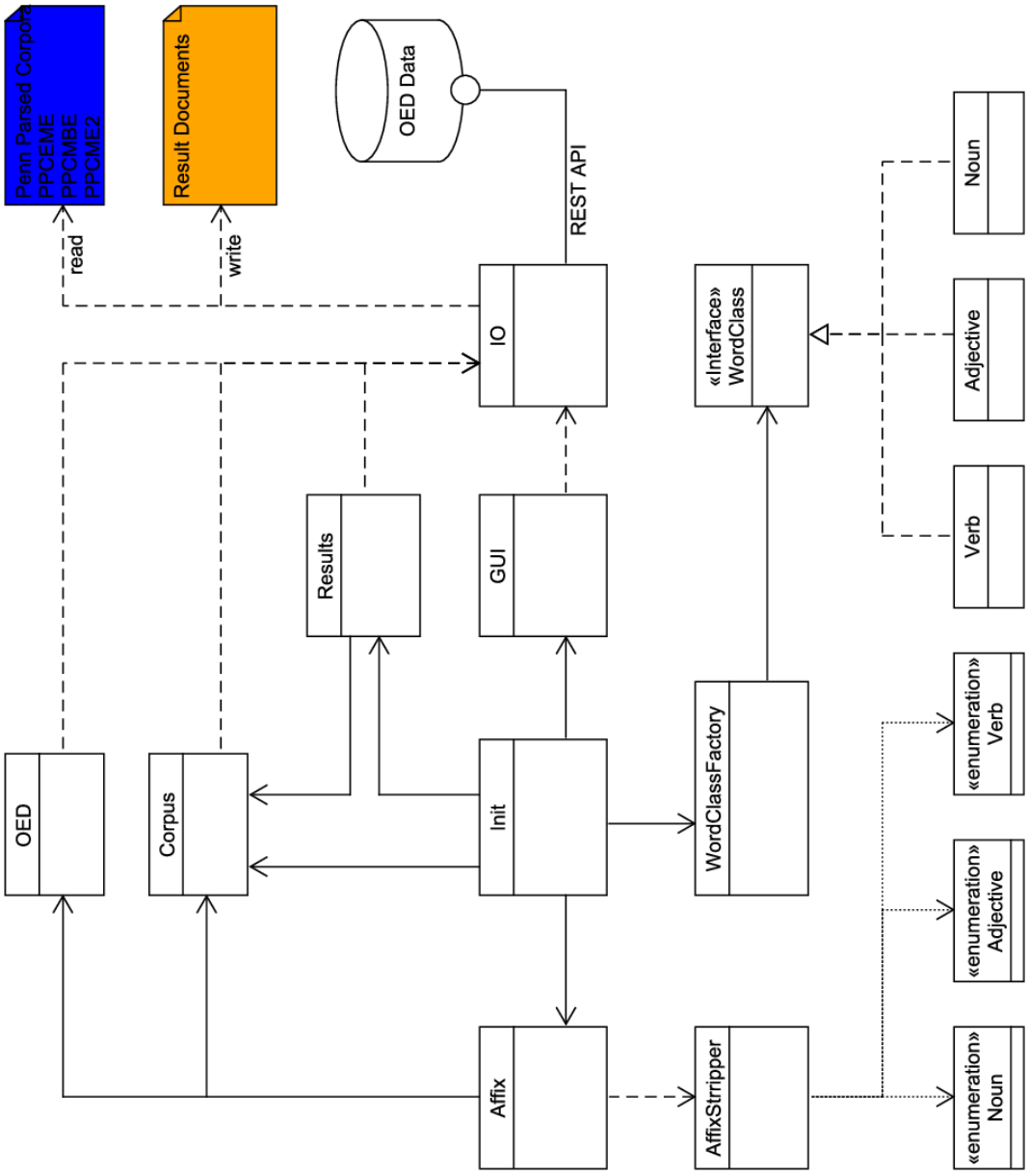

Figure 2: UML Class Diagram for Morphochron (4th generation)

Finally the fourth phase implemented the new RESTful API of the OED (see figure 2). It includes the main functionality of its predecessors although it is completely refactored. Generally it follows the idea known in information science as micro services. Once the OED decided to offer a functional API and make it available for research use free of any charge, the future way to go in using this resource was strikingly obvious. Because this approach successfully collected sufficient amounts of language data from a given text corpus and at the same time is the most advanced and exact method predominating the solution of Middle English affix extraction while incorporating some (but not all) of the previous algorithmic solutions, it is the prioritized candidate to be recommended for future use.

3. Conclusion

The above explications on a given (and long standing) problem in Diachronic Linguistics depicts a picture of how the existing inventory of AI-methods is typically applied. There are hardly any straight imperatives of proceedings that we could follow. It cannot be predicted with a higher or lower probability or plausibility as to which method fits better than the other. Indeed, it is possible to make a reasonable selection from the method inventory – i.e. exclude neural networks because the data does not fulfill its very basic requirements – but it still leaves the researcher with too many alternatives from which it is impossible to estimate a success rate. What seems to be an trial-and-error approach from the outside, it is a kind of systematic *polling* from the inside perspective. In the concrete case described here, one could learn that, on the one hand, the right combination from a semi-automatic method (first generation) enriched with a smart algorithm (2nd generation) would only be efficient if extended with a quality resource (4th generation). On the other hand, none of these components can be missed out, however, as the third generation showed, not all methods are equally optimal.

References

[1] H. Peukert, The Morphilo Toolset: Handling the Diversity of English Historical Texts, in: A. Ammermann, A. Brock, J. Pflaeging, P. Schildhauer (Eds.), Facets of Linguistics, Language and Text Studies, Peter Lang, Frankfurt, 2014, pp. 161–172. doi:10.3726/978-3-653-03540-7.

[2] M. Burghardt, C. Müller-Birn (Eds.), Merging Community Knowledge and Self-Interest to Build Language Resources: Architecture and Quality Management of a Take-and-Share-Approach of Word Annotations, Gesellschaft für Informatik e.V., Bonn, 2018. doi:10.18420/infdh2018-01.

[3] H. Peukert, Smoothing derivational asymmetries in English: In support of Greenberg's Universal 27, STUF - Language Typology and Universals 69 (2016) 517–545. doi:10.1515/stuf-2016-0022.

[4] H. Peukert, From semi-automatic to automatic affix extraction in Middle English corpora: Building a sustainable database for analyzing derivational morphology over time, in: J. Jancsary (Ed.), Proceedings of KONVENS 2012, ÖGAI, 2012, pp. 415–423. URL: http://www.oegai.at/konvens2012/proceedings/61_peukert12w/, lThist 2012 workshop.