Using a Hybrid Algorithm for Lemmatization of a Diachronic Corpus

Raoul Karimov[1[0000-0003-0313-0903]], ✉ Maria Samkova[1[0000-0001-6803-2701]], Svetlana Nikitina[1[0000-0002-1975-1256]], Andrei Akinin[1[0000-0001-5214-6819]]

[1] Chelyabinsk State University, Chelyabinsk, 454001, Russia
raoul.karimov@hotmail.com, maria.a.samkova@gmail.com, sanik09@list.ru, akinin96@gmail.com

Abstract. Lack of lemmatization often undermines the quality of concordances, which is especially relevant for diachronic corpora. A significant part of lemmatizers are designed for Modern English. This paper presents MiddleEnglishLem, an application designed for dictionary-based lemmatization of Middle English texts. We use a hybrid algorithm to lemmatize the Helsinki Corpus of English Texts, a long-time-span diachronic corpus that includes Middle English texts of different genres, a total of 608 570 words. MiddleEnglishLem is capable of associating multiple inflected forms and orthographic varieties with canonical forms. Lemmatization becomes more accurate owing to comprehensive premade dictionaries. The competitiveness of this lemmatizer is proved by the low average errors – less than 2.5 percent, whereas its prebuilt stemmer has a strength of 0.38, a relatively high value. Accuracy of the lemmatization process can be improved by implementing syntagmatic analysis at the part-of-speech identification step. MiddleEnglishLem can be applied to diachronic corpora in order to research the development of English.

Keywords: Concordance,· Lemmatization,· Diachronic Corpus,· Computer simulation,· Hybrid Algorithm,· Software Development.

1 Introduction

The authors of this paper have carried out an experiment with the diachronic part of the Helsinki Corpus of English Texts [1] to verify a glottochronology-based hypothesis. The hypothesis holds that diachronic changes of a vocabulary are predictable using a specific set of mathematical models. Such calculus-based modelling uses frequency rank tables [2]. The experiment is considered invalid as the original Helsinki Corpus of English Texts is not lemmatized, meaning that concordances list inflected forms as separate words, whereas we actually have to compare the frequency of superlemmas in two time-separated states of the vocabulary. This is where the issue of lemmatization arises, as the researchers are in need of better concordances. It has, however, been found out that there has been so far developed no algorithm for lemmatization of Middle English corpora. The existing lemmatizers/stemmers for Modern English texts are obviously inappropriate for this task due to significant mor-

pho-logical discrepancies between these two diachronically separate forms of the English language. It is therefore our additional objective to create a program that would allow to lemmatize the Helsinki Corpus of English Texts.

2 Classification of Algorithms. Overview of Existing Software

To begin the development of a proprietary algorithm for lemmatizing the Helsinki Corpus of English Texts, we first of all had to decide which algorithm suits our needs better and which existing software could probably be applied. V.A. Yatsko [3] states that inflected words can be stem-associated by means of simple stemming or lemmatization. Existing stemmers and their types can be classified as follows:

Table 1. The classification of stemmers

<table>
<tr><td colspan="2">Stemmers per Hull 1996 [4]</td><td colspan="3">Stemmers per Jivani 2011 [5]</td></tr>
<tr><td>Dictionary-based</td><td>Algorithmic</td><td>Truncating</td><td>Statistical</td><td>Mixed</td></tr>
<tr><td>Use a lexicon of lemmas</td><td colspan="2">Based on affix removal</td><td>N-Gram and alike</td><td>Inflectional/Derivational
Corpus-based
Context-sensitive</td></tr>
<tr><td>Require a constantly updated lexicon</td><td colspan="4">Make errors of two types:
Overstemming: different words stemmed to the same root
Understemming: words that should be stemmed to the same root are not</td></tr>
</table>

Lemmatization, according to Yatsko, differs from stemming in the approach to part-of-speech identification. Unlike stemmers of any type, lemmatizers identify parts of speech and take them into account when associating inflected forms with their respective lemmas. In the English language, words may be homographic yet belonging to different parts of speech, making lemmatization a considerably more reliable approach when it comes to concordance-building. Besides, algorithmic stemmers are actually designed to reduce multiple inflected forms to their stems, and stems are not always identical to canonical forms. Lemmatization, on the other hand, is always aimed at returning such forms and not just stems. Stemming is therefore considered a simplified alternative to lemmatization, which can be unsuitable for some research objectives. S.Th. Gries and A.L. Berez, however, mention that multiple stemming/lemmatization technologies can be combined to create a hybrid approach [6], which is going to be our case.

As of today, two popular stemmers used for English corpora are the Porter Stemming Algorithm by Martin Porter and the Lancaster Stemmer by Chris Paice and Garth Husk. While comparable in terms of stemming strength when applied to Modern English, none of these stemmers functions for Middle English, which is explicitly stated by Mr. Porter himself on his webpage. Yet capable of automatic normalization

of Early New English texts, i.e. correcting their orthography in accordance with the modern standards, the Porter stemmer will not cope with the complex morphology and non-codified writing of Middle English. The same applies to MorphAdorner, a popular lemmatizer which includes both the Porter stemmer and the Lancaster stemmer as importable modules. Thus, no piece of software we have been able to test could be used for our research, and an algorithm of our making has become a necessity.

3 Research Material: The Helsinki Corpus of English Texts

The experiment mentioned in the introduction used data extracted from the Helsinki Corpus of English Texts, which has a diachronic part covering the period of 730-1710, i.e. from late Old English till Early Modern English. The total volume of the corpus is about 1.5 million words (or word occurrences), which breaks into circa 450 texts belonging to philosophical, religious, scientific, fictional, educational and instructive writing as well as private correspondence. Dialectal division is present as well, with four dialects distinguished for Old English and five distinguished for Middle English. However, the main criterion used to group text samples together is their time of origins.

The Helsinki Corpus of English Texts uses the COCOA tagging standard and is therefore compatible with the Oxford Concordance Program. However, tagging is only used in corpora files to indicate some non-linguistic or extra-linguistic parameters of text samples, i.e. the date of creation, the date of manuscript-making, the author and their status, the dialect, etc. Within sentences, tags are used to distinguish Latin citations and editorial commentaries from the main text. That being said, the corpus as available via the Oxford Text Archives is neither syntactically parsed nor lemmatized. Due to the lack of lemmatization, the Oxford Concordance Program counts each inflected or orthographically different word form as a separate lexeme, which is why it returns incorrect statistical counts.

For our main experiment, we chose two diachronically separate sections of the corpus, Sections MEI and MEII (see [7] for the explanation of such choice). Therefore, we have a smaller corpus of approximately 210 thousand words to lemmatize, which is still a too significant amount, rendering any attempts of manual lemmatization unfeasible. For automatic lemmatization, we decided to use the same sections as experimental samples to check the actual functionality and appropriateness of our lemmatizing script. If properly programmed, it should return a lemma list that can be imported in the Oxford Concordance Program so that the latter can build appropriate and accurate concordances for the same sections.

4 Challenges of Middle English Lemmatization. Methodology

The linguistic nature of Middle English is very challenging when it comes to language processing; this paragraph is to analyze what kind of challenges we are facing and how we can cope with them while developing our lemmatization algorithm.

The first and most obvious challenge is the phenomenon of suppletion, i.e. use of "inflected forms" that do not share any stem with their canonical forms. For instance, we had such forms as *us, ur(e)*, which were not morphological derivatives of the dictionary form. Suppletion has existed since Proto-Germanic into Old English, changed a little in Middle English [8] and is present in Modern English as well, most strikingly in pronouns.

While manual lemmatization of suppletive forms is certainly not that difficult due to the small number thereof, this issue is further complicated by another peculiarity of Middle English, which is its non-codified orthography. Essentially, the graphical representation of many consonantal and vocalic clusters was not standardized until the 18th century, resulting in single words having multiple orthographic varieties. Those were not dialectal or even author-dependent, as even one text could contain, for instance, both *scylde* and *scilde* (these are the same word, *shield*). *C* could be written instead of *K* and vice-a-versa, *ou* and *u* were mutually replaceable as well, and the same applies to such clusters as *y/ge/i/ȝ/ghe* (the prefix of the Participle II form). Gradually, *th* came to replace letters "thorn" and "eth", but such replacement was not consistent until much later [9]. A simple solution would be to consider such mutually replaceable elements of writing as equivalent character combinations, but there might have existed some instances where the use of one such element had a differential effect. Non-codified orthography, however, is not limited to non-standardized use of some clusters; there are cases where versions of a single word are barely recognizable, e.g. *bringen*, the past (simple) form of which could be *abrouhte* and *bryggte*, further complicating any attempts at automatic processing of such texts.

Strong verbs represent another significant challenge, as some of their forms have the root vowel changed, e.g. the past (simple) plural form of *riden* is *roden*. It should be taken into account, however, that unlike weak verbs, strong verbs did not have dental suffixes as markers of their past forms, a fact that may help enhance the algorithm. The aforementioned participle II form of many verbs is another hereto related problem, as it often had a *y/ge/i/ȝ/ghe* prefix (a similar phenomenon is observable in Standard (or High) German of modern times). For instance, *y-sungen* was the Participle II for *syngen* [10]. Therefore, the morph-truncating algorithm should also include a verb-exclusive function to remove the prefix. This, however, may malfunction for verbs like *yelden*, where *y* is not a prefix but a part of the root.

The primary issue, however, is part-of-speech determination. On the one hand, Middle English was rather rich in terms of morphology, and nominal parts of speech had specific sets of morphs. On the other hand, many morphs did coincide for nouns and adjectives. Finally, even having a very accurate preset list of morphs will not enable appropriate differentiation of morphs and morph-like letter clusters. For instance, for *weorde*, which is a verb, *-de* is a suffix, but for *Franclonde*, which is a

proper noun, it is not. Therefore, part-of-speech tagging is not always possible by means of simple morph-truncating, meaning that the algorithm will require a preset dictionary listing already PoS-tagged lemmas.

So far, we have come to a simple yet labor-intensive solution to combine both truncating and dictionary-based stemming, resulting in the creation of a dictionary-dependent lemmatizer. This is a very conventional and somewhat obsolete approach, as modern algorithms mostly rely on finite-state transducers. However, we simply do not have sufficient volumes of data to train and refine an automaton-based machine. Besides, such an algorithm is far more difficult to develop, and we currently believe that even the manual preparation of a PoS dictionary makes more sense in our case. A dictionary like Mayhew and Skeat's [11] can be used to build such lemma dictionary ("the lexicon") listing both canonical forms and possible orthographic varieties. Using the lexicon will help distinguish nouns and verbs ending in similar letter combinations, like the aforesaid *Franclonde* and *weorde*. The program will therefore have a number of premade files, one of which will list possible morphs for the morph-truncating script, and others will list actual lemmas and their orthographic varieties. That being said, the program should be capable of:

- searching for isolated (tokenized) words from Middle English samples in the premade lexicon files;
- truncating morphs at a length of 1 to 3 symbols from the last character of a word;
- checking whether truncated morphs are found in the preset morph lists and whether they correspond to parts of speech as per the lexicon;
- correct association of inflected forms and lemmas and further registration of such associations in the output file, which is to be imported in the Oxford Concordance Program for further analysis of the Helsinki Corpus of English Texts;
- dealing with the strong verbs and overall inflectional peculiarities of verbs as a grammatical class;
- returning an error log containing all the words that could not be lemmatized for the subsequent manual lemmatization thereof.

The next paragraph describes the implementation of these functions.

5 Computational Implementation. Experiments

For the purpose of automatic lemmatization of Middle English texts, we have developed our own program titled MiddleEnglishLem using Python 2.7.9 as the programming language. We have chosen this language because it provides well-developed high-level data structures as well as a simple and efficient approach to object-oriented programming. Besides, Python is perfect for script-making and fast development of multi-platform applications for various purposes.

The application we have developed uses a set of input files, one of which is a plain text file that contains the corpus to process; another one is an .xslx file that enumerates morphs and their respective PoS properties; the rest files are PoS-specific tabular dictionaries, i.e. noun.xlsx, verb.xslx, etc., containing pre-associated lemmas and their orthographic varieties as listed in Mayhew and Skeat's (collectively, "the lexicon").

The output of the application is recorded in two separate files, one for lemmatized words and their forms/varieties and one for errors (words the algorithm cannot process).

The algorithm functions by simple morph truncation. It truncates a sequence of one symbol, adding one more if a single end symbol is not sufficient for processing. Truncated morphs are searched for in the morph set, the remainder of the token is searched for in the lexicon. As soon as the morph-associated part-of-speech tag matches that of the stem as indicated in the dictionary, the application records the token from the corpus in the tabulated output file under the lemma (if the latter is not present in the output file, it is copied from the lexicon). If the PoS tag is V (verb), the algorithm checks whether the participle II prefix is present and removes it. Verbal ablauts are not dealt with specifically as the lexicons list past tense forms for strong verbs. If the token cannot be processed after all steps are taken, it is written in the error log and skipped; the application then proceeds to the next token.

To test the efficiency of our script, we decided to use Zipf's law as an indicator of statistical representativeness:

$$f(k;s,N) = \frac{1}{k^s H_{N,s}} \tag{1}$$

where f is the relative frequency of a word in a corpus,

k is the rank of the word,

H is nth generalized harmonic number,

N is the number of words in the corpus,

s is the exponent value characterizing the Zipfian distribution of the text [12].

The exponent was found by least squares calculations in the R programming environment, where we used the *lzipf* package. It is believed that the value should be as closed as possible to 1 for natural languages. For the raw (unprocessed) text of the MEI subcorpus, it was 0.8546697, which does not meet the requirement above. Whether the word frequency distribution in the sample matched or did not match Zipf's law was verified by Pearson's chi-square test:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2}$$

where O_i is the total of all actual frequencies in the ith interval,

E_i is the total of projected (Zipfian) frequencies in the same interval [13]. For calculations, we divided the entire frequency table into 20 intervals with each subsequent interval being shorter under the harmonic principle; thus we had 18 degrees of freedom; at a 0.95 confidence interval, the chi-square distribution quantile for 18 degrees of freedom equals 28.8693. The chi-square value we obtained per formula (2) for the non-lemmatized sample was 68.5504 which meant that the sample did not match Zipf's law.

The sample was then lemmatized using MiddleEnglishLem; exclusive of Latin citations and proper names, the application returned 550 words, which amounted to circa 3% of the number of ranks in the pre-lemmatization frequency table. Lemmatization also reduced the number of such ranks by 34%, i.e. a third of the entire volume became associated with other lexemes in the table. The *s* value for Zipf's law was recalculated and found equal to 0.9625564. The chi-square was recalculated as well and equaled 26.52005. As this value was less than the quantile in our case, post-lemmatization word frequency distribution in the sample was confirmed to be in line with Zipf's law, a fact we believe proves that MiddleEnglishLem can help improve the statistical representativeness of MiddleEnglishTexts.

6 Conclusion: Unresolved Issues and Further Development

We have so far developed an algorithm allowing to lemmatize Middle English texts at a relatively low error rate; the built-in stemmer of our own making is considerably strong due to the natural morphological complexity and relatively poor vocabulary of Middle English. However, it will take more time and effort to prepare a full-fledged lexicon and apply the algorithm to the Helsinki Corpus of English Texts Middle English sections in their entirety. Besides, the algorithm still does not deal with some orthographic ambiguities of this language, i.e. it is not capable of recognizing character clusters with graphical varieties like *c*/*k* or *u*/*ou*. This may result in significant understemming, if such varieties are not included in the input lexicon files. On the other hand, some grammatical forms of different lexemes can be homographic, e.g. *fet* as a 3SGPresInd form of a verb, and *fet* as a noun. This issue, which in rare cases may lead to overstemming, can be solved by implementing syntagmatic analysis at the part-of-speech identification step, as non-functional parts of speech naturally tend to occur in certain syntactic structures [14]. The morphological complexity of verbs, especially strong verbs, is also a problem to be solved; we can further address the way it is dealt with in lemmatization algorithms for Standard German, where similar complexity exists. These three issues will be our main priorities when attempting to enhance the algorithm further.

One more challenge we are facing that will require very thorough analysis is the non-codified orthography of Middle English. While adding multiple orthographic varieties to the lexicon is a suitable solution, it means our program is only semi-automatic and still requires a lot of manual preparations. Use of finite-state transducers, a completely different approach, could be a solution to this problem if we had larger text samples for proper supervised machine learning. However, the approach will be discussed in further works.

References

1. Helsinki Corpus of English Texts, http://www.helsinki.fi/varieng/CoRD/corpora/HelsinkiCorpus/

2. Arapov, M.V., Herz, M.M.: Mathematical Methods in Historical Linguistics. Nauka, Moscow (1974). (in Russian).
3. Yatsko, V.A.: Algorithms and programs for automatic text processing. Bulletin of Irkutsk State Linguistic University, vol. 1 (17), pp. 150–161 (2012).
4. Hull, D.A.: Stemming algorithms: a case study for detailed evaluation. Journal of the American Society for Information Science, vol. 47, N 1, pp. 70–84 (1996). doi: 10.1002/(SICI)1097-4571(199601)47:1%3C70::AID-ASI7%3E3.0.CO;2-%23
5. Jivani, A.G.: A Comparative Study of Stemming Algorithms. International Journal of Computer Technology and Applications, vol. 2 (6), pp. 1930–1938 (2011).
6. Gries, S.Th., Berez, A.L.: Linguistic annotation in/for corpus linguistics. In: Nancy Ide & James Pustejovsky (eds.), Handbook of Linguistic Annotation. Berlin & New York: Springer (2015).
7. Karimov, R.D.: Predictive Modelling of the Development of Middle English vocabulary. Linguistics and Translation Issues Studied by Young Scientists, Nizhny Novgorod, vol. 1, pp. 189–198 (2013). (in Russian).
8. Hogg, R.: An Introduction to Old English. Edinburgh University Press, Edinburgh (2012).
9. Ward, A.W., Waller, A.R.: Changes in the Language to the Days of Chaucer: Middle English Spelling. In: The Cambridge History of English and American Literature in 18 Volumes, vol. 1. From the Beginnings to the Cycles of Romance. Cambridge University Press, Cambridge (1907).
10. Ilyish, B.A.: History of the English Language. Vysshaya Shkola, Moscow (1968). (in Russian).
11. Mayhew, M.A., Skeat, W.: A concise dictionary of Middle English from A.D. 1150 to 1580. Clarendon Press, Oxford (1888).
12. Moreno-Sánchez, I., Font-Clos, F., Corral, A.: Large-Scale Analysis of Zipf's Law in English Texts. PLoS ONE (2016). doi:10.1371/journal.pone.0147073
13. Preacher, K. J.: Calculation for the chi-square test: An interactive calculation tool for chi-square tests of goodness of fit and independence [Computer software] (2001). Available from http://www.quantpsy.org/chisq/chisq.htm
14. Sidorov, G., Velasquez, F., Stamatatos, E., Gelbukh, A., Chanona-Hernández, L.: Syntactic Dependency-Based N-grams: More Evidence of Usefulness in Classification. CICLing 2013. Part I. LNCS 7816, pp. 13–24 (2013). doi: 10.1007/978-3-642-37247-6_2