Readme

The alignment method for baseline.py is: pad the original word and inflections until find the alignment that has the lowest hamming distance (highest overlapping) e.g. excecate_ _:excecating, acetize_:acetizes, atray_ _: atrayed and save prefix rules($ > $ge) and suffix rules($cetize: $cetizes) to corresponding MSD dictionary. Next time when a new word comes out for inflection generation: check MSD and look into corresponding rule collections 1. For prefix, add the most frequent one 2. For suffix, add the longest suffix conversion and if there is a tie, use the frequency for selection.

To improve the results from the algorithm, we decided to add some additional rules which were not covered for the two languages we chose - English and German. This algorithm takes into account rules which are language-specific and which would have been applied by humans, if transformed manually.

For English:

- Some words that are supposed to change from ‘consonant + y’ to ‘consonant + ied/ies’ are ignored by the current algorithm. Examples found are ajaxify: ajaxified, endamnify : endamnified, viddyed : viddied, verbifys : verbifies, pussify : pussified. To address this, we hard code the command for if a “consonant + y” is seen at the end of the word, we will apply a conversion to “consonant + ies/ied”
- Some words have mistakenly doubled the last letter (quitclaimming: quitclaiming, backburnerred: backburnered while some should be doubled are not doubled (bedrumed : bedrummed, excured : excurred, cross-refering : cross-referring. The current algorithm only captured the doubling rules sometime but can’t differentiate scenario that this rule should be applied to with a limited training corpus. To address this, we hard code the command for implementing last letter doubling: if a “consonant1 + vowel + consonant2” is seen at the end of the word, we will apply a conversion to “consonant1 + vowel + consonant2 + consonant2 + ed/ing”,
- ‘ss/chs’ —-> ‘sses’ is not counted in the current method(overdosss: overdosses) To address this, we hard code the command: if a “ss/ch/sh/x/zh” is seen at the end of the word, when changing to 'V;3;SG;PRS' we will add “es” instead of the most common “+s”.

For German:

- The majority of the incorrect outputs were because some of the prefixes which should be split from the verb in certain forms weren’t accounted for, e.g. ausscheiden -> sheiden aus, einteil -> teil ein, etc.
- Related to this, some verbs had incorrect past tense form, where the ge- prefix was at the wrong place. In German, the ge- should come in between the prefixes that split and the stem of the verb, not at the very beginning. Additionally, we shouldn’t have ge- if the words starts with one of the prefixes which do not get split, or if we have a verb ending with -iert (in past tense), e.g. gebemuttet -> bemuttet.

- Finally, for all words with the label N;ACC;SG we shouldn't see a change between the original and outform.

Our method improved the results slightly:
english[task 1/high]: 0.954
german[task 1/high]: 0.842
Average[high]: 0.898

However, if this task is done completely by scripting the rules per language, as it was done in previous methods, it is possible that we do not achieve results better than the algorithm. One thing we noticed, however, was that the algorithm only performs better if we have a lot of data which is why it's not very good at low resource languages. Similarly, if we do have a lot of data but too many various grammatical label options, the performance is also not good. Additionally, the algorithm cannot deal with exceptions which have relatively low occurrences in the corpus very well, while exceptions are often seen in many languages.