Initial Thoughts on Gvkey Matching

Task: Firm name matching, accompanied by gvkey matching

Problem: The same firm can have similar but non-identical names.

Datasets:

- Compustat – containing firm name, Gvkey, tickers and country (The set of variables in datasets might have changed since the last pull).
- Hassan's Firm-Level Political Risk Dataset – containing firm name and Gvkey.

Current algorithm:

- Pre-processing stage:
    - Process firm names: lowercase, delete all special symbols, and delete all common words[1].
- Exact matching stage:
    - Check for exact matches using tickers.
    - The unmatched firm names remain.
- Fuzzy matching stage:
    - Use compare-function to return Pr(strings are equal) based on Jaro string distance.
    - First match firm names with the same ticker and keep the match if Pr(strings are equal) > 0.8.
    - Then match remaining firm names with all possible names and keep the match if Pr(strings are equal) > 0.9.

Ideas for improving the current algorithm:

- Improve on the pre-processing stage with some substitutions, etc.
- Improve on the exact matching stage
    - Get some other data that would help identify the firm and reduce the number of firm names that need to undergo fuzzy matching.
    - Feature engineering, e.g. get the canonicalized name
- Improve on the fuzzy matching stage
    - Use some other definition or combination of string similarity / string distance (e.g. Levenshtein distance, cosine similarity, ROUGE-L) with the current fuzzy name matching algorithm.
        - ROUGE-L: https://www.tensorflow.org/text/tutorials/text_similarity
        - Fuzzywuzzy (Levenshtein distance): https://pypi.org/project/fuzzywuzzy/
        - Possibly a combination of these metrics

[1] ["earnings conference call","conference call on productivity", "earnings release conference", "financial release conference","conference call regarding", "earnings conference", "comprehensive review", "final transcript", "edited transcript", "week conference", "conference call", "edited brief", "preliminary brief", "earnings call", "earning call", "preliminary transcript", "final transcript", "call","cal","merger","c", "earning","earnings", "to discuss","group","plc","ltd","limited","ag","corp","corporation","Incorporation","laboratories","labs","the","propo sed","propose", "holdings","oyj","inc","conference","co", "final","preliminary","and","&", "company","trust","investment","investments","sln","sa","s.p.a.","spa","transc", "quarter","st","nd","rd","th", "q", "jan","feb","mar","apr","may","jun","jul","aug","sep","oct","nov","dec"]

- Use some other fuzzy name matching algorithm (examples below, in no particular order):
    - Non-ML methods
        - Word embeddings: https://rcpedia.stanford.edu/topicGuides/textProcessingWord_Embeddings.html
        - N-grams, Term frequency-inverse document frequency (TF-IDF) https://towardsdatascience.com/surprisingly-effective-way-to-name-matching-in-python-1a67328e670e
        - Phonetic algorithms (Soundex, Metaphone, Double Metaphone): https://towardsdatascience.com/python-tutorial-fuzzy-name-matching-algorithms-7a6f43322cc5
    - ML methods
        - Ensemble: https://medium.com/bcggamma/an-ensemble-approach-to-large-scale-fuzzy-name-matching-b3e3fa124e3c