Seth Cattanach
CSE-44657 Natural Language Processing
Final Project Report
Due: Monday, 22 April 2019

I. Goals

The Healthcare Lemmatizer is a lemmatization tool that uses a semi-supervised machine learning approach to process text in the healthcare/patientcare domain. It aims to expand WordNet's capabilities by training a model on healthcare-specific terms, and using the identified lemmatization rules, lemmatize words that do not exist in the WordNet database.

Such a tool has several advantages, namely that it extends the benefits of lemmatization to the healthcare domain. Normalizing text, such as doctor's notes from patient visits, allows for more efficient – and more accurate – text processing, because a script will have fewer unique tokens to analyze after lemmatization has been performed. One example of a relevant application could be creating a machine learning algorithm to predict likelihood of patient mortality based on patient notes (text) from a recent hospital visit.

II. Methods

The project includes four parts:

1) The Python program **generate.ipynb** parses patient notes from a clinical database (MIMIC-III) and creates a list of tokens that could not be lemmatized by the baseline method, WordNet Lemmatizer
2) I then created a "gold standard" of 200 lemma mappings from this list of tokens; the list, **lemma-mappings.txt**, is included in the source code, and a sample is provided below. Each comma-separated term is a distinct lemma mapping with the format **('token', 'part-of-speech'):'lemmatized form'**

    ('portosystemic', 'a'):'portosystemic', ('hexavitamins', 'n'):'hexavitamin',

    ('subsalicylate', 'n'):'subsalicylate', ('reevaluated', 'v'):'reevaluate',

3) The Python program **train.ipynb** trains a model to identify rules based on these "gold standard" lemma mappings. I created a semi-supervised machine learning algorithm to train this model; the algorithm primarily looks at word endings (minimum of two characters) and creates a weighted list of lemmatization rules. Rules are weighted by how often they occur, so the rule [verb that ends with '-ed' → change suffix to '-e'] would be given additional weight for each time it occurs in the "gold standard" dataset. The specific rules are stored in a list called "rules" in this program. The rules are written to a model, **model.txt**, using Python's "pickle" library to pack and unpack this model as a list.

4) Lastly, the Python program **test.ipynb** uses this model to lemmatize any tokens that could not be lemmatized by the WordNet Lemmatizer. The algorithm chooses the highest-weight (e.g. most frequently occurring) rule out of all the rules that apply to a specific token and lemmatizes the token by applying this rule. For example, for the token ['resuscitated', 'v'] the algorithm would identify the '-ed' ending + 'verb' part-of-speech tag and apply '-ed' → '-e' to lemmatize this token to 'resuscitate'. If there are no rules that apply to a given token, the algorithm does not try to lemmatize the token (this occurs most commonly due to misspelled words).

There are two similar approaches to the challenge of lemmatizing healthcare text. The WordNet project has a lemmatizer that uses a lookup-based approach; if a token is found in its database, it lemmatizes the token based on the mapping provided. This method is highly accurate (limited only by the accuracy of the lemma mappings in its database), but not very robust: an unidentified token is never lemmatized, and instead, the WordNet Lemmatizer simply returns the token unchanged. The WordNet Lemmatizer is an open-source project, and I used it as a baseline to evaluate my project in terms of lemmatization percentage. The source code can be found here: https://www.nltk.org/_modules/nltk/stem/wordnet.html

Another tool, the BioLemmatizer (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3359276/), is more similar to my own approach. This tool uses a lookup-based approach (similar to WordNet) for tokens that already exist in its database but uses a rule-based approach for words that do not exist, thus providing additional domain coverage. It was trained on a corpus of biomedical literature and tested on a biomedical corpus called *LLL05*. I could not get access to this corpus, so I did not evaluate my method compared to the BioLemmatizer tool, though I suspect it would be more accurate and robust than my tool due to the greater size of its training set and the domain expertise of the developers (where as my tool is limited by several factors, such as inaccurate part-of-speech tagging and the accuracy of my "gold standard" dataset). This tool is geared more toward biomedical text and healthcare research, whereas my healthcare lemmatizer was trained on *patientcare* text – a subtle but important difference.

III. Experiments

Using the MIMIC-III database (specifically, the "NOTEEVENTS.csv" table, which includes patient notes for ICU stays over a multi-year period), I trained a model on the first 1,000 patient notes (approximately 700,000 tokens after preprocessing). Using this model, I evaluated my healthcare lemmatizer in terms of lemmatization percentage (the percentage of tokens that were lemmatized, regardless if the lemmatized form was the same as the original token and regardless of whether the lemmatization was accurate) and accuracy (the percent of lemmatized tokens that were lemmatized correctly by my healthcare lemmatizer). The lemmatization percentage was compared to the baseline method (WordNet Lemmatizer).

Five trials were conducted, each on a separate subset of patient notes. These testing sets were unique from the training set; the training set covered patient notes 1-1000, and the five evaluation trials each used the next 1000 notes (trial 1 used notes 1001-2000, trial 2 used notes 2001-3000,

etc.). The lemmatization percentage of my algorithm compared to WordNet Lemmatizer is reported below in **Table 1**.

Trial	1 [1000:2000)	2 [2000:3000)	3 [3000:4000)	4 [4000:5000)	5 [5000:6000)	AVG
Number of tokens	686,037	669,144	701,146	686,627	667,744	682,140
Not found by WordNet Lemmatizer	40,968	40,345	43,022	42,175	40,173	41,337
Not found by Healthcare Lemmatizer	17,991	17,489	19,078	18,512	17,217	18,057
Lemmatized %: WordNet	**94.0%**	**94.0%**	**93.9%**	**93.9%**	**94.0%**	**93.9%**
Lemmatized %: Healthcare Lemmatizer	**97.3%**	**97.4%**	**97.3%**	**97.3%**	**97.4%**	**97.4%**

Table 1: % of tokens lemmatized by WordNet Lemmatizer (baseline) and the Healthcare Lemmatizer

Using the same five trials, the accuracy of my healthcare lemmatizer was also calculated. This percentage only represents tokens that my algorithm lemmatized but WordNet did not; that is, all the additional tokens lemmatized by my algorithm. The accuracy of these five trials is reported in **Table 2**.

Trial	1 [1000:2000)	2 [2000:3000)	3 [3000:4000)	4 [4000:5000)	5 [5000:6000)	AVG
Correct	714	671	796	703	705	718
Incorrect	28	20	20	17	16	20
Accuracy	**96.2%**	**97.1%**	**97.5%**	**97.6%**	**97.8%**	**97.3%**

Table 2: Lemmatization percentage of my algorithm on expanded domain of tokens

A note about data: I used distinct subsets of the MIMIC-III database to train and evaluate my algorithm. This database contains real patient data, including physician notes, mortality outcomes, demographic information, etc. and is protected by PhysioNet. I cannot release the data and so it cannot be found in my GitHub repository with the rest of my project. However, there are two options to replicate this study; please see **V. Replicability** (below) for further discussion.

IV. Conclusions

The healthcare lemmatizer outperformed the baseline method, WordNet Lemmatizer, significantly in terms of lemmatization percentage; it was able to lemmatize an additional ~2.5% of tokens, proving it could provide expanded domain coverage for a dataset of healthcare and patientcare text.

My healthcare lemmatizer also provided a high level of accuracy, on average lemmatizing over 97% of unseen tokens correctly. I did noticed that many of the incorrectly-lemmatized tokens

tended to follow the same patterns; for example, the token “reformatted” was always (incorrectly) lemmatized to “reformatte” due to the high weight of a rule that lemmatizes verbs ending in ‘-ed’ to ‘-e’ (whereas the correct lemmatized form, in this case, should have been “reformat”).

One of the biggest limitations of accuracy and robustness/coverage is part-of-speech tagging. During my initial trials, I noticed the NLTK tagger was much more accurate than the WordNet tagger. However, the WordNet lemmatizer requires all POS tags are provided in its own format (‘n’=noun, ‘v’=verb, ‘a’=adjective, etc.) where as NLTK tags words with over 30 possible options (‘NN’=singular noun, ‘NNS’=plural noun, ‘NNP’=singular proper noun, etc.). Since part-of-speech tag accuracy is vital to the performance of my algorithm, I used the more accurate NLTK tool and modified an open-source list of mappings to map NLTK tags to their WordNet equivalents. For more information, see the comments in **generate.ipynb**.

My algorithm’s accuracy and coverage was also limited by the accuracy of my “gold standard” dataset (since I’m not a healthcare domain expert, the lemma mappings I created by hand – which are used by the machine learning algorithm – are likely not entirely accurate) as well as the size of my dataset (though I had over two million patient notes and likely over a billion total tokens in the MIMIC-III database, due to time considerations, I only created 200 lemma mappings in my “gold standard” database; more mappings would likely increase both the accuracy and coverage of my algorithm).

V. Replicability

The source code and necessary text files to replicate this project are all included in my GitHub repository (https://github.com/scattana/healthcare-lemmatizer) with one exception – the MIMIC-III database (which includes the NOTEEVENTS.csv table) cannot be included due to privacy restrictions of patient data (more information: https://physionet.org/physiobank/database/mimic3cdb/)

There are two options to resolve this issue:

1) Apply for access to the MIMIC-III database and download the NOTEEVENTS.csv table. This process requires completing several short online modules and applying for access through PhysioNet (see link above), but it also requires a legitimate research reason and advisor recommendation/sponsorship.
2) Replicate the NOTEEVENTS.csv table with fake data using the following columns and datatypes, where **TEXT** is the patient note for that particular hospital admission:

   ROW_ID [int], **SUBJECT_ID** [int], **CHARTDATE** [datetime], **CHARTTIME** [datetime], **TEXT** [string]

   (the *datetime* type has the following format: [**2008-06-14**] 06:22 = June 14, 2008 6:22 AM