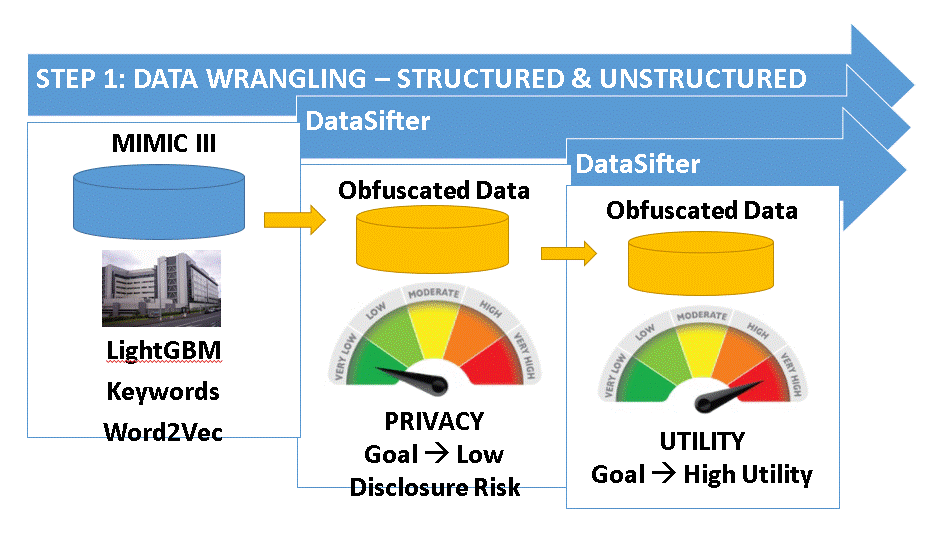
DataSifter Workflow

**Step 0**: Can we work on a schematic cartoon that lays out the entire workflow?? Below is a snapshot of something I sketched out in PowerPoint. Let’s generate a great image !! Who will take the lead on this?

**Step 1**: The data we use includes structured numerical/categorical columns and unstructured text

Possible dataset to analyze:

* MIMIC3 data used in the White Paper for predictive analytics. The outcome to predict was Length of Stay. we can use that as the “sensitive outcome” of Step 2. NOTE: we need to retrieve Doctor Notes for each SubjectID in the dataset used in the White Paper (unless they are available in another dataset that we already retrieved). We will then have a dataset with structured and unstructured/text data types
* Join the result from white paper and the original MIMIC 3 using ”SUBJECT\_ID”
* Dataset created by Yunze and will be shared on Slack (dedicated channel) and/or GitHub
* The dataset will have ~3,500 unique records with 36 features, one of each text/doctor notes

**Step 1.1: Run the code that “cleans up” the data**

<https://colab.research.google.com/drive/1-Qsk-SjeZAVUM6geB6ACSxRQ30Rjip8k#scrollTo=dd8cde0e> [this might not be everything, but we can add to it]

**Step 2**: Identify sensitive outcome(s) to protect

* Length of stay (continuous)
* Religion (categorical with >5 levels)
* Gender or any other binary outcome

**Step 3** [Yiming/Yunze/Chengfan]: LightGBM on the text feature/cell to identify predictive keywords for the sensitive outcomes in Step 2 separately.

* Run as many LightGBM as many outcomes listed in step 2
* Generate a list of W keywords (W=10, as a suggestion)

**Step 4** [Yitong]: Build a “semantic radius” around each keyword to establish a semi-quantitative distance metric to anchor different levels of obfuscation. For example, the top 10 keywords for the outcome “Marital Status” are: 'wife', 'husband', 'married', 'alone', 'daughter', 'widowed', 'son', 'lives', 'sex', 'she'. Based on semantic meanings, we can extract two “semantic clusters”: 1: 'wife', 'husband', 'daughter', ‘son’; 2: 'married', 'alone', 'widowed'. This step is to guarantee the readability of obfuscated text.

Input 1 (for obfuscation): # keywords (higher = more obfuscation)

Input 2 (for obfuscation): radius around each keyword using word2vec (further away = more obfuscation)

For example:

*small*=top 3 keywords, 0-33% distance from each keyword

*medium*=top 5 keywords, 34-66% distance from each keyword

*large*=top 10 keywords, 67–100% distance from each keyword

# keyword=k

# words on the semantic radius=N

Extra columns/features for SDV = k + k\*N = k\*(N+1)

**NOTE: the training can be done on the same data we ran LightGBM OR on a larger dataset with more semantic options [google….]**

RECIPES FOR STEP 5

**FIRST SOLUTION**

* For each keyword generate the top 100 word with word2Vec
* Sample 1 word randomly from each of the subsets below
* The first 33 words will be the pool for the small obfuscation level
* The 34th-66th words will be the pool for the medium obfuscation level
* The 67th-100th words will be the pool for the large obfuscation level
* KEYWORD: wife → 'husband', 'married',[0-33%] 'alone', 'daughter', 'widowed',[34-66%] 'son', 'lives', 'sex', 'she'[67-100%] **YUQI/YUNZE ARE WORKING ON THIS !!**
* **We can use an empirical distribution to assess the %s**
* *semantic\_radius(keyword,#words on the radius=N)*
* word2vec\_small(*keyword*=wife, *semantic\_radius=*search words within dist of 2 from wife,1 to *N/3*)
* word2vec\_medium(*keyword*=wife, *semantic\_radius*=search words within dist of 2-4 from wife,N/3 to 2N/3)
* word2vec\_large(*keyword*=wife, *semantic\_radius=*search words within dist of 4-6 from wife,2N/3 to N)

N words on the spectrum small-medium-large

**Yuqi, please give an example of your current Python implementation.**

word2vec: <https://en.wikipedia.org/wiki/Word2vec#:~:text=Word2vec%20is%20a%20technique%20for,words%20for%20a%20partial%20sentence>. Training on the current data in order to vectorize - DATA DEPENDENT

GloVe: <https://en.wikipedia.org/wiki/GloVe_(machine_learning)> Training on the current data in order to vectorize - DATA DEPENDENT

BERT: <https://en.wikipedia.org/wiki/BERT_(language_model)> UNIVERSAL

**Step 5 [not done yet]**: Apply Datasifter to obfuscate Data: swap the keywords within “semantic clusters”, may use BERT to further improve readability?

After this step, we will have 4 datasets with doctor notes/text:

* Original
* Small obfuscation
* Medium Obfuscation
* Large obfuscation

**Step 6** [Yitong]: recast text (before and after obfuscation) into tabular (e.g., numerical matrix)

Text → 2d matrix with each row is a word, each column is the Word2Vec dimension, which is variable (~100 or 200). Numerical values in the columns are not the distance.

**Step 7 [not done yet]:**

**Privacy metric**: apply SDV package to the recasted datasets (before and after obfuscation)

**Utility metric**: apply LightGBM on the text feature/cell to identify predictive keywords for the sensitive outcomes in Step 2 and after Step 5 (after semantic radius has been ap[plied and keywords swapped).

Basic workflow:

<https://colab.research.google.com/drive/1UzIoWJbm03IeRW4vZf6tIjMY4yUm7cmA#scrollTo=ttPhNAjYa5if>

<https://github.com/SOCR/DataSifterText2>

Step 1: pre-processing of the text (Python code)

Step 2: select outcome of interest Y

Step 3: run LightGBM ⇒Y=f(original/cleaned text)

Step 4: 10 keywords returned by Step 3

**Step 5: replace keywords in the original text with the criterium described above (please confirm how we do this)**

Step 6: repeat Step 5 to generate 3 clones of the original/cleaned text (X) - small/medium/large (

Step 7: use word2vec to represent each cell with text as a numerical matrix with as many rows as words and ~100 columns with values taken from the first 100 elements of word2vec representation

Step 8: replace the 3 “text” clones and the original/clean text in Step 6 with numerical clones (X1, X2 and X3 numerical clones, where X is the original numerical text)

Step 9: apply sdv functions g() as Privacy-small = g(X, X1), Privacy-medium = g(X, X2), Privacy-large = g(X, X3).

Step 10: Step 9 returns **Privacy metrics** comparing only text. We can apply g() to the remaining data that are numerical/categorical.

**Step 11: define Utility metrics** (see below)

1. OPTION 1 (probably not feasible)
   1. Run Step 3 with Y=f(text-clone-small) , Y=f(text-clone-medium) , Y=f(text-clone-large)
   2. Compare the top 10 keywords generated across the 3 lightGBM vs Step 4 keywords
   3. Overlap % indicates utility (this should be all 0%, because we swapped all the keywords)
   4. In b we could select >10 keywords and compare to the list in Step 4 (where we saved >10, but only used the top 10 for obfuscation)
2. OPTION 2 (probably not feasible)
   1. Run a **predictive model** (GLM–DL) with the numerical clones of the original/cleaned text Y=f(X), Y=f(X1) , Y=f(X2) , Y=f(X3)
   2. Compare the prediction (or parameter estimates) across the 4 models as measure of utility
3. THINK OF OPTION 3 !!!
   1. ….
   2. ….