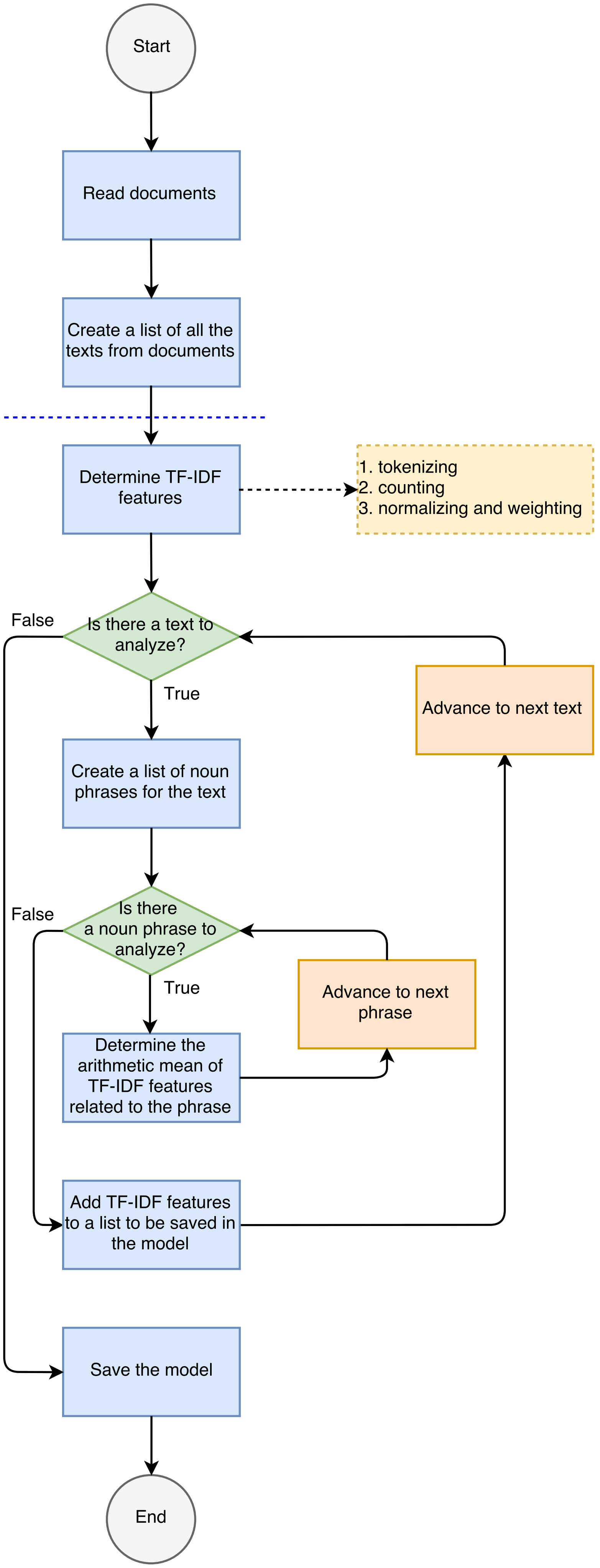
Hybrid System 1: Construction and Training of the Model

Source Code

https://github.com/nasa-jpl/ASSESS/blob/master/webapp/text_analysis/model.py

```
build_system1(texts_all,n=2):

tfidftransformer=TfidfVectorizer(ngram_range=(1,n))
texts_all_tf=tfidftransformer.fit_transform(texts_all)

master_phrases_vectors=[]
for text_tf,text in zip(texts_all_tf,texts_all):

phrases=use.noun_tokenize(text)
phrases=list(set(phrases))
phrases_vectors=[list(tfidftransformer.transform([x])[0]
                      .indices) for x in phrases]

phrases_dict={}
for x,phrase in zip(phrases_vectors,phrases):

x=np.array(text_tf).flatten()[x]
avg=np.mean(x)
phrases_dict[phrase]=avg

master_phrases_vectors.append(phrases_dict)

use.savemodel(master_phrases_vectors,
              'master_phrases_vectors_1')
use.savemodel(texts_all_tf,'texts_all_tf_1')
use.savemodel(tfidftransformer,'tfidftransformer_1')
```

Flow Chart



Start
Read documents
Create a list of all the texts from documents
Determine TF-IDF features
1. tokenizing
2. counting
3. normalizing and weighting
Is there a text to analyze?
False
True
Create a list of noun phrases for the text
Advance to next text
Is there a noun phrase to analyze?
False
True
Determine the arithmetic mean of TF-IDF features related to the phrase
Advance to next phrase
Add TF-IDF features to a list to be saved in the model
Save the model
End

Data Structures and Comments

texts_all `["text_1", "text_2", "...", "text_n"]`

texts_all_tf

	term_1	term_2	term_3	...	term_m
doc_1					
doc_2					
...					
doc_n					

unigrams (**n=1**) and bigrams (**n=2**)

`tf-idf(t,d) = tf(t,d) * idf(t)`

where idf is computed as

$$\mathrm{idf}(t) = \log \frac{n_d}{\mathrm{df}(d,t)} + 1$$

where n_d is the total number of documents, and df(d,t) is the number of documents that contain term t.

text `"text_<i>"`

text_tf

	term_1	term_2	term_3	...	term_m
doc_<i>					

phrases `["noun_phrase_1", "noun_phrase_2", "..."]`

phrases_vectors `[[index_1_1, index_1_3, ...], [index_2_1, index_2_2, ...], ...]`

x `["index_<i>_1", "index_<i>_2", "..."]`

phrase `"noun_phrase_<i>"`

x `["tfidf_1", "tfidf_2", "..."]`

phrases_dict `[["phrase_1", avg_1], ["phrase_2", avg_2], ...]`

master_phrases_vectors

```
[[["phrase_1", avg_1], ["phrase_2", avg_2], ...],    ← doc_1
 [["phrase_1", avg_1], ["phrase_2", avg_2], ...],    ← doc_2
 [...],                                              ← ...
 [["phrase_1", avg_1], ["phrase_2", avg_2], ...]]    ← doc_n
```