Problem Statement

It is uncertain if Natural Language Processing Techniques can be used to automate the identification of risks from protocols as the foundation for the Adaptive Monitoring Assessment Process.

In [1]:
```
from __future__ import print_function

from docx import Document
from docx.shared import Inches

import re
import nltk
from nltk import tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize, sent_tokenize
import spacy
import mglearn

from sklearn.feature_extraction.text import  CountVectorizer
from sklearn.decomposition import  LatentDirichletAllocation
from collections import Counter
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
import random
import numpy as np

import sys
if not sys.warnoptions:
    import warnings
    warnings.simplefilter("ignore")
```

Get Segment

In [22]:
```
def getSegment(doc, heading):
    #heading = 'Inclusion criteria'
    document = Document(doc)
    i = -1
    st = 0
    en=0
    seg_text =  ''
    for para in document.paragraphs:
        i += 1
        if para.text == heading:
            st = i + 1
            inc_sty = para.style
        if st > 0:
            if para.style == inc_sty and i > st:
                en = i
                break

    for para in document.paragraphs[st:en]:
        seg_text += para.text


    return seg_text
```

In [23]:
```
def getAllText(doc):
    #heading = 'Inclusion criteria'
    document = Document(doc)
    i = -1
    st = 0
    en=0
    seg_text =  ''
    for para in document.paragraphs:
        seg_text += para.text


    return seg_text
```

In [40]:
```
text = getSegment('protocols/Immu09.docx','Inclusion Criteria')
#text = getAllText('protocols/Immu09.docx')

alltext = getAllText('protocols/Immu09.docx')
```

In [ ]:

In [26]:
```
def sentence_tokens(itext):
    #mask all dots between numbers
    pattern = re.compile(r'(?<=\d)[.](?=\d)')
    isatext = pattern.sub('_isadot_',itext)

    #prepare sentence for tokenization
    isatext = isatext.replace(':', '. ').replace('\t', ' ').replace('.', '. ')

    sent_text = nltk.sent_tokenize(isatext)

    sent_text1 = []
    for sen in sent_text:
        sent_text1.append(sen.replace('_isadot_', '.'))

    return sent_text1
```

In [43]:
```
incld_list =  sentence_tokens(text)
protocol_list = sentence_tokens(alltext)
```

Latent Dirichlet Allocation Summarization

In [195]:
```
n_samples = len(incld_list)
n_features = round(n_samples * 10)
n_topics = 20
n_top_words = 20

def print_top_words(model, feature_names, n_top_words):
    list_term_temp=[]
    #list_idx=[]

    for topic_idx, topic in enumerate(model.components_):
        #list_term_temp=[]
        #print("Topic #%d:" % topic_idx)
        #list_idx.append(topic_idx)
        #print(" ".join([feature_names[i]
        #                for i in topic.argsort()[:-n_top_words - 1:-1]]))
        for i in topic.argsort()[:-n_top_words -1:-1]:
            list_term_temp.append(feature_names[i])

        #list_term.append(list_term_temp)

    #dic=pd.DataFrame({'topic_index':list_idx, 'terms':list_term})
    #print()
    return list_term_temp
```

In [146]:
```
n_features
```

Out[146]:
```
510
```

In [196]:
```
data_samples_incld = incld_list

tf_vectorizer_incld = CountVectorizer(max_df=0.8, min_df=1, ngram_range=(2,2),
                                max_features=n_features,
                                stop_words='english')

tf_incld = tf_vectorizer_incld.fit_transform(data_samples_incld)
```

In [197]:
```
pd.DataFrame(tf_incld.toarray(),columns=tf_vectorizer_incld.get_feature_names())
```

Out[197]:

	000 mm3note	10 days	100 000	12 consecutive	12 month	12 months	120 days	14 days	17 beginning	18 years	...	usual lifestyle
0	0	0	0	0	0	0	0	0	0	1	...	0
1	0	0	0	0	0	1	0	0	0	0	...	0
2	0	0	0	0	0	0	0	0	0	0	...	0
3	0	0	0	0	0	0	0	0	0	0	...	0
4	0	0	0	0	0	1	0	0	0	0	...	0
5	0	0	0	0	0	0	0	0	0	0	...	0
6	0	0	0	0	1	0	0	0	0	0	...	0
7	0	0	0	0	0	0	0	0	0	0	...	0
8	0	0	0	0	0	0	0	0	0	0	...	0
9	0	0	0	0	0	0	0	0	0	0	...	0
10	0	0	0	0	0	0	0	0	0	0	...	0
11	0	0	0	0	0	0	0	0	0	0	...	0
12	0	0	0	0	0	0	0	0	0	0	...	0
13	0	0	0	0	0	0	0	0	0	0	...	0
14	0	0	0	0	0	0	0	0	0	0	...	0
15	0	0	0	0	0	0	0	0	0	0	...	0
16	0	0	0	0	0	0	0	0	0	0	...	0
17	0	0	0	0	0	0	0	0	0	0	...	0
18	0	0	0	0	0	0	0	0	0	0	...	0
19	0	0	0	0	0	0	0	0	0	0	...	0
20	0	0	0	0	0	0	0	0	0	0	...	0
21	0	0	0	0	0	0	0	0	0	0	...	0
22	0	0	0	0	0	0	0	0	0	0	...	0
23	0	0	0	0	0	0	0	0	0	0	...	0
24	0	0	0	0	0	0	0	0	0	0	...	0
25	0	0	0	0	0	0	0	0	0	0	...	0
26	0	0	0	0	0	0	0	0	0	0	...	0
27	0	1	0	0	0	0	0	0	0	0	...	0
28	0	0	0	0	0	0	0	0	0	0	...	0
29	0	0	0	0	0	0	0	0	0	0	...	0
30	0	0	0	0	0	0	0	0	0	0	...	0
31	1	0	1	0	0	0	0	0	0	0	...	0
32	0	0	0	0	0	0	0	1	0	0	...	0
33	0	0	0	0	0	0	0	0	0	0	...	0

	000 mm3note	10 days	100 000	12 consecutive	12 month	12 months	120 days	14 days	17 beginning	18 years	...	usual lifestyle
34	0	0	0	0	0	0	0	0	0	0	...	0
35	0	0	0	0	0	0	0	0	0	0	...	0
36	0	0	0	0	0	0	0	0	0	0	...	0
37	0	0	0	0	0	0	0	0	0	0	...	0
38	0	0	0	0	0	0	0	0	0	0	...	0
39	0	0	0	0	0	0	0	0	0	0	...	0
40	0	0	0	0	0	0	0	0	0	0	...	0
41	0	0	0	1	0	0	0	0	0	0	...	0
42	0	0	0	0	0	0	1	0	0	0	...	1
43	0	0	0	0	0	0	0	0	0	0	...	0
44	0	0	0	0	0	0	1	0	0	0	...	0
45	0	0	0	0	0	0	1	0	0	0	...	0
46	0	0	0	0	0	0	0	0	1	0	...	0
47	0	0	0	0	0	0	0	0	0	0	...	0
48	0	0	0	0	0	0	0	0	0	0	...	0
49	0	0	0	0	0	0	0	0	0	0	...	0
50	0	0	0	0	0	0	0	1	0	0	...	0

51 rows × 510 columns

In [204]:
```
print("Fitting LDA models with tf features, "
      "n_samples=%d and n_features=%d..."
      % (n_samples, n_features))

lda = LatentDirichletAllocation(n_topics=n_topics, max_iter=5,
                                learning_method='online',
                                learning_offset=50.,
                                random_state=0)

lda_incld = lda.fit(tf_incld)

tf_feature_names_incld = tf_vectorizer_incld.get_feature_names()
incld_terms = print_top_words(lda_incld, tf_feature_names_incld, n_top_words)

incld_terms_list = list(dict(Counter(incld_terms).most_common(20)))

unique_incld = list(set(incld_terms_list))
#unique_excld = list(set(excld_terms_list) - set(incld_terms_list))
```

```
Fitting LDA models with tf features, n_samples=51 and n_features=510...
```

In [205]:
```
unique_incld
```

Out[205]:
```
['human chorionic',
 'recently available',
 'local treatment',
 'performance status',
 'effective method',
 'days prior',
 'bone metastases',
 'measures dimension',
 'anticancer therapy',
 '14 days',
 'lesion measures',
 '12 months',
 'method contraception',
 'toxicities grade',
 'kinase inhibitor',
 'normal uln',
 'neuropathy grade',
 'group ecog',
 'study drug',
 'growth factor']
```

In [200]:
```
sorting=np.argsort(lda.components_)[:,::-1]
features=np.array(tf_vectorizer_incld.get_feature_names())
```

In [202]:
```
import mglearn
dd1 = mglearn.tools.print_topics(topics=range(5), feature_names=features,sorting
```

```
topic 0
--------
resolution systemic
disease allowed
bony disease
target lesion
systemic anticancer


topic 1
--------
screening labs
surgery stereotactic
stereotactic surgery
factor support
14 days


topic 2
--------
method contraception
function instead
plus diaphragm
alp uln
metastases case


topic 3
--------
measures dimension
lesions external
short axis
serum albumin
count 100


topic 4
--------
performance status
upper limit
days prior
group ecog
oncology group
```

Type *Markdown* and LaTeX: α^2

PDFMIner - [https://www.binpress.com/manipulate-pdf-python/ (https://www.binpress.com/manipulate-pdf-python/)](https://www.binpress.com/manipulate-pdf-python/)

BeautifulSoup - https://www.dataquest.io/blog/web-scraping-tutorial-python/ (https://www.dataquest.io/blog/web-scraping-tutorial-python/)

PyTextRank - https://medium.com/@aneesha/beyond-bag-of-words-using-pytextrank-to-find-phrases-and-summarize-text-f736fa3773c5 (https://medium.com/@aneesha/beyond-bag-of-words-using-pytextrank-to-find-phrases-and-summarize-text-f736fa3773c5)

Text Summarization with NLTK in Python - https://stackabuse.com/text-summarization-with-nltk-in-python/ (https://stackabuse.com/text-summarization-with-nltk-in-python/)

Text summarization in 5 steps using NLTK - https://becominghuman.ai/text-summarization-in-5-steps-using-nltk-65b21e352b65 (https://becominghuman.ai/text-summarization-in-5-steps-using-nltk-65b21e352b65)

TFIDF - https://towardsdatascience.com/tfidf-for-piece-of-text-in-python-43feccaa74f8 (https://towardsdatascience.com/tfidf-for-piece-of-text-in-python-43feccaa74f8)

NLP For Topic Modeling Summarization Of Financial Documents https://blog.usejournal.com/nlp-for-topic-modeling-summarization-of-financial-documents-10-k-q-93070db96c1d (https://blog.usejournal.com/nlp-for-topic-modeling-summarization-of-financial-documents-10-k-q-93070db96c1d)

This is a nice subject to play with LDA on! It might also be cool to see how treating individual sentences as documents could affect topics. Computationally more expensive, but it might be feasible.

https://towardsdatascience.com/basic-nlp-on-the-texts-of-harry-potter-topic-modeling-with-latent-dirichlet-allocation-f3c00f77b0f5 (https://towardsdatascience.com/basic-nlp-on-the-texts-of-harry-potter-topic-modeling-with-latent-dirichlet-allocation-f3c00f77b0f5)

In [98]:

```

```

Out[98]: 9