Activités du Lundi 18 mars au Vendredi 22 mars :

- **Utilisation de Stanford Parser avec NLTK (from nltk.parse.stanford import StanfordParser)**
  - Transformation des fichiers TXT de TimeBank :

```
[Tree('ROOT', [Tree('NP', [Tree('NN', ['ABC19980108']), Tree('CD', ['.1830.0711'])])])][Tree('ROOT',
[Tree('S', [Tree('NP', [Tree('NP', [Tree('DT', ['The']), Tree('JJ', ['financial']), Tree('NN',
['assistance']), Tree('NN', ['#e3'])]), Tree('PP', [Tree('IN', ['from']), Tree('NP', [Tree('NP',
[Tree('DT', ['the']), Tree('NNP', ['World']), Tree('NNP', ['Bank'])]), Tree('CC', ['and']), Tree('NP',
[Tree('DT', ['the']), Tree('NNP', ['International']), Tree('NNP', ['Monetary']), Tree('NNP',
['Fund'])])])])])]), Tree('VP', [Tree('VBP', ['are']), Tree('RB', ['not']), Tree('VP', [Tree('VBG',
['helping']), Tree('NP', [Tree('NNS', ['#e4'])])])]), Tree('.', ['.'])])])][Tree('ROOT', [Tree('S',
[Tree('PP', [Tree('IN', ['In']), Tree('NP', [Tree('NP', [Tree('DT', ['the']), Tree('JJ', ['#t85']),
Tree('JJ', ['>']), Tree('JJ', ['last']), Tree('NNS', ['#t85'])]), Tree('VP', [Tree('VBG', ['>']),
Tree('NP', [Tree('CD', ['twenty']), Tree('JJ', ['#t85']), Tree('JJ', ['>']), Tree('CD', ['four']),
```

Stanford Parser ne peux pas parser et tagger tout le texte en une seule fois sinon nous obtenons une erreur java du type out of memory :

```
**********************************************************
***  WARNING!! OUT OF MEMORY! THERE WAS NOT ENOUGH  ***
***  MEMORY TO RUN ALL PARSERS.  EITHER GIVE THE    ***
***  JVM MORE MEMORY, SET THE MAXIMUM SENTENCE      ***
***  LENGTH WITH -maxLength, OR PERHAPS YOU ARE     ***
***  HAPPY TO HAVE THE PARSER FALL BACK TO USING    ***
***  A SIMPLER PARSER FOR VERY LONG SENTENCES.      ***
**********************************************************

Sentence has no parse using PCFG grammar (or no PCFG fallback).  Skipping...
Exception in thread "main" edu.stanford.nlp.parser.common.NoSuchParseException
        at edu.stanford.nlp.parser.lexparser.LexicalizedParserQuery.getBestParse(LexicalizedParserQuery.java:403)
        at edu.stanford.nlp.parser.lexparser.LexicalizedParserQuery.getBestParse(LexicalizedParserQuery.java:375)
        at edu.stanford.nlp.parser.lexparser.ParseFiles.processResults(ParseFiles.java:272)
        at edu.stanford.nlp.parser.lexparser.ParseFiles.parseFiles(ParseFiles.java:216)
        at edu.stanford.nlp.parser.lexparser.ParseFiles.parseFiles(ParseFiles.java:75)
        at edu.stanford.nlp.parser.lexparser.LexicalizedParser.main(LexicalizedParser.java:1518)
```

Donc j'ai appliqué Stanford Parser pour chaque phrase de chaque texte. Nous avons donc tous nos fichiers sous le format précédemment observé dans un nouveau dossier.

Cependant, ce format-là ne convient pas comme input à addDiscourse (pour la récupération des signaux qui ne sont pas annotés dans TB). addDiscours attend un input de ce genre :

```
(S1 (S (NP (NN Selling)) (VP (VBD picked) (PRT (RP up)) (SBAR (IN as) (S (S (NP (JJ previous) (NNS buyers))
(VP (VBD bailed) (PRT (RP out)) (PP (IN of) (NP (PRP$ their) (NNS positions))))) (CC and) (S (NP (JJ
aggressive) (JJ short) (NN sellers--anticipating)) (VP (ADVP (RB further)) (VBD declines--moved) (PRT (RP
in)))))))) (. .)))
(S1 (S (NP (PRP$ My) (JJ favorite) (NNS colors)) (VP (AUX are) (ADJP (JJ red) (CC and) (JJ green))) (. .)))
(S1 (S (S (NP (NP (DT The) (NNS asbetsos) (NN fiber)) (, ,) (NP (NN crocidolite)) (, ,)) (VP (AUX is) (ADJP
(RB unusually) (JJ resilient)) (SBAR (IN once) (S (NP (PRP it)) (VP (VBZ enters) (NP (NP (DT the) (NNS
lungs)) (, ,) (PP (IN with) (NP (NP (RB even) (JJ brief) (NNS exposures)) (PP (TO to) (NP (PRP it))) (VP (VBG
causing) (NP (NP (NNS symptoms)) (SBAR (WHNP (WDT that)) (S (VP (VBP show) (PRT (RP up)) (ADVP (NP (NNS
decades)) (RB later))))))))))))) (, ,) (NP (NNS researchers)) (VP (VBD said)) (. .)))
```

  - Il faut donc transformer, avec des expressions régulières, les données obtenues par Stanford Parser pour qu'addDiscourse puisse les analyser.
  - Problème détecté sur ne premier fichier, le stanford parser ne prend pas en compte la première phrase (à voir si il le fait pour tous les fichiers)

- **Utilisation de addDiscourse**
  - En ligne de commande dans python :

```
for filename in os.listdir(path_input):
    file = path_input+filename
    os.system(path_addDiscourse+' --parses '+file+' --output '+path_output+filename)
```

- On passe en input les fichiers obtenus par Stanford Parser et en output on obtient le même format que Stanford Parser mais avec les signaux annotés.
- « it takes syntactic parse trees as input and outputs augmented trees with tags for each discourse connective »
- Types de connecteurs :
  - Temporal
  - Expansion
  - Contingency
  - Comparison
  - 0 (non-discourse usage)

- **Reconstruction du corpus**
  - Le but est de retirer toutes les marques de Stanford Parser (parenthèses, pos tagging, …) afin de récupérer le texte comme il l'était avant de le transformer en « arbre syntaxique plat ». La suppression de ces caractères se fait avec des expressions régulières. (A finir)
  - Les fichiers obtenus après reconstruction deviendra notre nouvel input dans le programme dataframe.py et extractFeatures.py (partie où les fichiers txt sont utilisés)

Après la reconstruction du corpus, il faudra assigner un identifiant artificiel (sid) à chaque signal obtenu. Enfin, tout sera prêt pour intégrer les signaux dans les csv, à la suite des signaux d'AQ.