Activités du Lundi 25 février au Vendredi 1er mars :

- **Création d'un nouveau CSV pour les signaux (docId, sid, libellé du signal)**

- **Modification des POS « unknown » dans le fichier csv_features_events.csv**

Lors de la tâche de récupération de la première lettre des POS des events pour les lemmatiser avec NLTK, nous avons remarqué que dans le corpus AQUAINT, nous pouvons trouver de nombreux events dont le POS est annoté par « unknown ».

Mon script python permettait de lemmatiser avec la première lettre du POS si le POS n'était pas « unknown ». Et si le POS était « unknown », il le restait. Nous avons donc remarqué que dans le CSV apparaissaient beaucoup de « unknown » dans la colonne des lemmes.

Dans le but d'améliorer la colonne « lemmeNltk », Anaïs m'a fait remarquer que si nous avons un POS= « unknown » alors nous devions regarder la colonne « tense ». Si le temps était « None » alors il était fort probable que nous trouvions un event dont le POS est un nom -> « NOUN », sinon si tense est différent de « None » alors il fallait prendre par défaut le POS « VERB ».

```
if pos == 'unknown':
   if tense == 'NONE':
      pos = 'noun'
   else:
      pos = 'verb'
```

- **Création des fichiers textes sans balises**

Dans le but de simplifier la structure XML des fichiers texte ont été créés. Ils contiennent la transformation des fichiers XML mais sans les balises. Les events sont marqués par event#eid exemple said#e1. De même pour les timex : timex#tid et les signaux : signal#sid.

Les timex avec plusieurs lexèmes prennent cette forme : a#t1 minute#t1 and#t1 a#t1 half#t1. Ils ont le même tid car font partie du même timex.

Nous avons un problème dans les fichiers csv, lorsque les timex ne comprennent qu'un seul mot sont détectés (comme « Friday », s'ils en ont plus ils n'étaient pas trouvés (comme « a minute and a half »). Cette méthode d'associer un même identifiant à tous les éléments d'un même timex pourrait nous aider à résoudre ce problème. De même si un event comprend plusieurs unités lexicales.

- **Découpage du programme extractFeatures.py en fonctions**
    - Fonction d'ouverture des fichiers
    - Fonction d'extraction des instances / des events / des timexs / des signaux
    - Fonction pour lemmatiser les events
    - Fonction pour écrire dans les différents fichiers CSV (csv_features_timex, csv_features_event, csv_features_signal)
    - …

- **Recherche sur comment utiliser addDiscourse**

Nous avons remarqué que dans le corpus TB les signaux (after, before, on,…) ne sont pas annotés. addDiscourse est un outil (script en perl) qui nous permettra de les détecter.

addDiscourse annote de cette manière (n° = identifiant + type du connecteur) :

as → as#6#Temporal

In addition → In#0#Expansion addition#0#Expansion. (Marquage des unités multi-mots grâce à l'identifiant)

Téléchargement de addDiscourse http://www.cis.upenn.edu/~nlp/software/discourse.html
Readme : http://www.cis.upenn.edu/~nlp/software/addDiscourseREADME.txt

En ligne de commande : addDiscourse.pl **--parses** sample-parse.txt **--output** sample-out.txt

En entrée, addDiscourse.pl attend un fichier étiqueté syntaxiquement.
Un étiquetage comme ci-dessous :
(S1 (S (NP (NN Selling)) (VP (VBD picked) (PRT (RP up)) (SBAR (IN as) (S (S (NP (JJ previous)

Nous devons donc trouver un étiqueteur qui nous donne cet étiquetage.

- **Stanford Parser** semble adapté. Les fichiers ont été téléchargés mais il faut faire plusieurs manipulations avant que le parser fonctionne dans python.

- **Récupération du contexte des events :**

Changement de méthode :

- Utilisation des fichiers .txt (avec les event#eid / timex#tid / signal#sid).
- Problème : **nltk.word_tokenize tokenize** tous les symboles et ponctuation.
  Pour régler ce problème j'ai utilisé la librairie **TweetTokenizer** de Nltk qui permet, si nous avons #abc, de garder la séquence entière, sans la tokenizer en '#', 'e1'.
  Donc nous avons par exemple : 'said', '#e1'.
  Ensuite j'ai trouvé un algorithme qui permet de rassembler les deux pour avoir '**said#e1**'.