Synthèse

Problème majeur du sujet

Présenter la complexité du problème

Le problème principal rencontré est dû à la variété et diversité du langage. (cf. art.(6))
La classification d'uttérances - ne reposant pas sur la présence de mots clé, d'attributs ou de features spécifiques en raison d'un thème prédominant - ne permet pas vraiment une approche de type *sentiment analysis, classification bayésienne*.

Le manque de structure des sentences et leur manque d'ordre apparent (chaotique) ne préconise pas ce genre d'approche.

D'autre part, une variété d'approches sont proposées dans les papiers lus. (différents algorithmes d'apprentissage, utilisation de tagging et structures grammaticales, d'outils complémentatires comme SEANCE..etc)

Pourquoi approche word embedding peut être intéressante ?

Justifier l'approche

A documenter davantage, il s'agit du travail en cours.

Une approche word embedding permettrait de représenter les sentences sous forme de vecteurs.

Cette approche semble prendre en compte un mot et son contexte, ce qui pourrait permettre d'éviter le cas par cas sur les mots présents relativement à un vocabulaire restreint.

La notion d'espace sémantique – permettant de rapprocher les mots par leur sens et de les comparer – permettrait de généraliser le vocabulaire et de ne pas se restreindre à la présence ou non de mots en particulier.

Une approche qui paraît relativement moderne.

L'approche semblait également s'inscrire dans la continuité de la réflexion décrite lors de la présentation du sujet.

Comment le réaliser ?

Présenter les outils possibles

Certains outils déjà existant permettraient une telle mise en œuvre.

La librairie SpaCy est dotée de méthodes permettant le preprocessing lexical (tokenisation, lemmatisation, stopwords, utilisation de labels...etc) ainsi qu'une représentation vectorielle des sentences avec l'utilisation du word embedding (.vector, .similarity … etc). (cf. liens (1))

SpaCy, avec l'utilisation également de gensim, permet également l'utilisation de models tels que word2vec ou Glove.

Méthode

Présenter un raisonnement possible

Méthode générale

1. Récupérer la donnée du fichier

La récupération des données du fichier est réalisée avec la librairie Panda.
Plusieurs fichiers ont été générés, présentant la structure des données et les regroupant par classes.

2. Preprocessing

La classe preprocessing.py a été implémentée, permettant le preprocessing des utterances. Cette classe contient des fonctions permettant la normalisation, la tokenisation, la suppression de stopwords, la lemmatisation.

Il est également possible d'utiliser les labels (~ tags) pour l'identification des personnes (prénoms) dans les dialogues. (ceci pourrait permettre de répondre à la problématique soulevée que des noms propres ont tendance à augmenter la distance entre deux utterances)

Une fonction n-grams peut également être ajoutée afin de tester une approche par n-grams. (approche qui a également été proposée avec monogrammes, bigrammes, trigrammes)

3. Représentation vectorielle

La structure de classe word_embedding.py est encore à définir.

Pour l'instant, les fonctions .vector ainsi que .similarity ont été expérimentées sur des tokens ainsi que sur des sentences, permettant de faire émerger plusieurs approches envisageables décrites dans le paragraphe suivant.

Cependant, l'utilisation de .similarity est accompagnée d'un warning :

> ModelsWarning: [W007] The model you're using has no word vectors loaded, so the result of the Doc.similarity method will be based on the tagger, parser and NER, which may not give useful similarity judgements. This may happen if you're using one of the small models, e.g. `en_core_web_sm`, which don't ship with word vectors and only use context-sensitive tensors. You can always add your own word vectors, or use one of the larger models instead if available.

Impliquant possiblement l'utilisation de models pour la représentation vectorielle des utterances.

L'utilisation de gensim peut être envisagée afin d'exploiter des models déjà existants tels que word2vec ou Glove. (à voir)

Une plus large documentation sur le word embedding sera nécessaire. Il s'agit du travail actuellement entrepris.

4. Algorithme de classification se basant sur la représentation vectorielle

utterance – vecteur – categorie

- KNN, clustering

La représentation en vecteurs implique des valeurs continues, ce qui peut permettre l'application de divers algorithmes de classification.

Approches
Plusieurs approches sont possibles pour la vectorisation :

- utterance : représenter sous forme de vecteur l'utterance complète et calculer la distance 'sémantique' entre ces utterances. Approche équivalent à « dans quelle mesure le sens des phrases se rapproche »

- Tokens : utiliser les stopwords, la tokénisation et la lemmatisation pour faire émerger un vocabulaire représentatif de l'utterance traitée. Chaque token est associé à un vecteur (valeur, espace sémantique). L'utterance est réprésentée par cette suite de tokens, c'est à dire la somme des vecteurs ?

(mise en œuvre nécessite davantage de documentation)

Autres propositions :

- utilisation des LABELS SpaCy pour identifier les personnes – prénoms

- utilisation d'outils préexistants :

p.ex : SEANCE, sentiment analysis (cf. art. (1))

Articles, références

Les références, articles lus jusqu'à ce jour

Articles :

(1) Combining Click-Stream Data with NLP Tools to Better understand MOOC Completion
(2) Time and Semantic Similarity – What is the Best Alternative to Capture implicit Links in CSCL conversation ?
(3) I say, You say, We say : Using Spoken Language to Model Socio-Cognitive Processes during Computer-Supported Collaborative Problem Solving
(4) PolyCAFe : Collaboration and Utterance Assesment for Online CSCL Conversations
(5) Utterance Classification in Auto Tutor
(6) Semantic Sentiment Analysis of Twitter

(7) Intent Classification for Dialogue Utterances

Livres :

(1) *Natural Language Processing an Computational Linguistics*, Bhargav Srinivasa-Desikan
(2) *Le Machine Learning avec Python*, Andreas C. Müller et Sarah Guido, O'Reilly
(3) *Big Data et Machine Learning,* Pirmin Lemberger, Marc Batty, Médéric Morel, Jean-Luc Raffaëli, Dunod

Liens :

(1) *https://spacy.io/usage/vectors-similarity*

Ce qui est à faire

- Documenter word embedding (connaissance encore insuffisante)

- Documenter les méthodes SpaCy .vector, .similarity : sur quoi se basent-elles ? Sont-elles suffisantes ?

- Lire les papiers trouvés sur **dblp** :

  - *Word Embeddings and their use in sentence classification tasks*,

    Amit Mandelbaum, Adi Shalev

  - *Efficient estimation of Word Representations in Vector Space,*

    Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dcan

  - *Distributed Representations of words and phrases and their compositionality,*

    Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dcan, Ilya Sustekever

  - Linguistic regularities in continuous space word representations,

    Tomas Mikolov, Wen-tau Yih, Geoffrey Zweig

  - *Sentiment analysis with contextual embeddings and self attention,*

    Kataryna Biesialska, Magdalena Bicsialska, Henryk Rybinski