Sentence clustering

Pertinence of Natural Language Processing method

When aiming at groups words and/or sentences, many approaches can be considered, including the manual one where a human simply reads out all the said words or sentences and groups them according to its own interpretation of what good groupings might be. In the case of a large number of items or long sentences, this method is hardly feasible. Instead, automatic approaches like the Natural Language Processing (NLP) ones seem more natural with their proposed computed solutions.

Beyond its automatic nature and quickness, the NLP method also has the advantage of being able to classify future words or sentences to the best computed grouping on previous words or sentences.

An important flaw of the NLP approach is that there is no automatic metric describing what a good grouping is or is not. For instance, most NLP methods generate vectorial representations of words based only on which words they are most often seen with. While this is pertinent, it lacks the presence of semantic or contextualized information for words whose meaning changes drastically depending on the context. As will be detailed below, this can lead to sentence encodings focusing more on syntactic content of a sentence like the presence of a negative tone rather than the actual subject of the sentence, on which we would most likely prefer to base our clustering.

Experiment settings

For each item and domain in the mobility datasets, we first apply a **word filtering**, followed by a **embedding mapping** of word groups to vectors. A **dimensionality reduction** is then applied on the resulting vectors which we finally feed to a k-means clustering algorithm with a predefined target **number of clusters**. All bold values are settings for which we explored different configurations. We describe them all further below.

A first setting considered was to filter some words hypothesized to only add noise to the word groups. The different filters considered are (1) the absence of filter, (2) filtering all words with less than 4 letters and (3) filtering words contained in a stop-words public dictionary.

From a filtered word group we now needed to extract vectorial representations. This is done by using publicly available word-level vectorial mappings. The word-level mappings used are GLoVe with 50, 100, 200 and 300 dimensional embeddings. We also experimented with the sate-of-the-art BERT model, both on word and sentence-level encodings. Lastly, we collected a total of 15 mobility-related articles, on which we trained our own FastText embeddings.

The dimensional reduction portion of the approach uses a Principal Component Analysis decomposition of the original sentence embeddings. The output dimensions explored are 2, 5, 10, 25 and keeping the raw embeddings.

For the number of clusters to look for by our algorithm, we used values ranging from 4 to 8 included.

Results

Results are given in the form of a .csv file containing items and domains grouped by clusters. The clusters are listed in order of proximity to their respective clusters, meaning that the top word groups of each column should be strong indicators of their clusters' contents.

The best results were all obtained by the BERT model, wether via sentence-level or word-level encoding. Obtained clusters appear much better than on the first iteration, with cluster sizes much more similar and cluster headers more descriptive of their cluster's content.

For both the domains and items clustering we present the few best results our approach yielded according to our own human interpretation. For all retained configurations, we include

- a `clusters.csv` file, containing the proposed groupings;
- a `config.txt` file, detailing which configuration was used;
- `pca_clusters.png` and `tsne_clusters.png` images, presenting 2 dimensional views of the used sentence embeddings.

We first note the very significant difference in sentence embeddings dispersion (shown in the `png` images) when using the BERT model in comparison with the previous iteration. The embeddings are much less grouped, allowing the clustering algorithm to more efficiently retrieve good groupings.

Moreover, we denote a much improved tightness of each cluster. This is particularly clear in the domains clusters, where the column headers can almost directly be used as cluster tag.

For the items clustering, the results are harder to grasp, mainly due to the very high number of sentences. The cluster sizes are in a satisfyingly similar range. The resulting clusters are however more difficult to analyze. Most clusters appear more linked on the syntactic similarity than semantic. For instance, in the 7 clusters retained grouping, the seventh cluster clearly contains all the sentences where the group 'how often' is present, no matter what the rest of the sentence might be about. A similar problem also arises with a cluster being focused on the presence of negation in the sentence.

Possible upgrades

We consider the domains clustering mostly done, with results satisfying enough to keep as is.

Regarding the items clustering task, we might consider another word filtering method: frequent word groups removal. We could automatically identify word groups that are often present and should not be used to group sentence together like *'how often do you'* or *'not at all'*. For instance, we could take the sentence *'how often do you go to parties'* and only use the *'go to parties'* part for the sentence embeddings. The effort to implement this particular upgrade would be in the analysis and collection of the said often seen and useless word groups.