CATS Usage Instructions

Prof. Dr. Goran Glavaš

June 2019

1 Introduction

CATS (**C**oherence-**A**ware **T**ext **S**egmentation) is a state-of-the-art tool for text segmentation. At the core of the tool is a deep neural model for text segmentation based on hierarchical two-layer Transformer architecture. CATS enables coherence-aware text segmentation, as it was trained in a multi-task learning setup, joining segmentation prediction with coherence modelling. The auxiliary coherence modelling is what gives CATS the edge over competing segmentation methods and leads to state-of-the-art segmentation performance.

This document provides instructions on how to use the CATS software to automatically segment textual documents.

2 Quick Start

CATS comes with pre-trained segmentation models that can be immediately used to segment texts. In the CATS directory, one can directly use the BASH script `segment.sh` to segment texual documents in a given folder.

```
bash segment.sh [-s 0|1] [-p 0|1] input_dir output_dir
```

Argument `input_dir` is the path to the directory that contains textual documents that need to be segmented. Argument `output_dir` is the path to the directory in which the corresponding segmented documents will be stored. The option `-s` indicates whether the input documents, to be segmented, are already sentence-segmented (i.e., in the one-sentence-per-line format; value `1`) or not (if they are just raw text without sentence segmentation, the value of the `-s` flag should be set to `0`). The option `-p` indicates if the segmentation probability predictions will be written for each sentences (value `0`) or not (value `1`). The default value for both options `-s` and `-t` (i.e., if not provided) is `0`.

The script `segment.sh` merely sequentially executes two Python scripts:

- `cats_preprocess.py` – converts the raw textual documents into data structures (concretely, TensorFlow Records) consumed by the pre-trained

neural segmentation models; Upon completion, this script (temporarilly) generates two special files – `records.tf` and `blocks.pkl` – in the `output_dir`. These serialized data structures are then the input for the second script;

- `cats_predict.py` – generates segmentation predictions (takes as input `records.tf` and `blocks.pkl` generated by `cats_preprocess.py`) and creates segmented variants of the input files. The segmented documents are saved to the `output_dir`. After the segmented textual documents have been generated, the script `segment.sh` deletes the temporary serialization files (`records.tf` and `blocks.pkl`) generated by `cats_preprocess.py`.

3 Tool Configuration

The CATS tool is configurable via the configuration file `config.py`. The following are the descriptions of the most important configuration sections.

Data Section. In this section of the configuration file, we mostly define the paths to the relevant data files (pre-trained word embeddings, serialized training/development/test files, etc.). The following are the most relevant configuration variables:

- `texts_lang`: specifies the language of the input texts. The default language is English (`en`), but CATS can also segment texts in a number of other languages: German (`de`), Italian (`it`), French (`fr`), Spanish (`es`), Czech (`cs`), Croatian (`hr`), Russian (`ru`), Finnish (`fi`), and Turkish (`tr`). CATS can easily be extended to additional languages – all that is needed is to project the pre-trained word embeddings of a new language to the provided English embedding space (given in `data/embeddings/en.vectors`) and add the corresponding vectors and vocabulary files to the subdirectory `data/embeddings`;

- `seg_start`: specifies the string that (in its own line) indicates the start of the new segment. This is relevant for (1) preprocessing raw text files with gold-annotated segmentation (using the script `cats_preprocess.py`) – in this case the value of `seg_start` must exactly match the string that is used to denote the segment start in the documents; (2) outputing segmented files: the string specified with `seg_start` will be used before the first sentence of each predicted segment;

- `fake_sent`: This is just a dummy sentence that is used to pad the sequences of sentences to the standard length used in the model training. No need to change or adjust this value;

- `vocab_path_en` and `vecs_path_en`: A (relative) path to the pre-trained English embeddings (vocabulary file and vectors file, respectively). The

paths are already relatively set to the pre-trained English fastText vectors. Unless you want to plug in some other pre-trained embeddings (e.g., word2vec or GloVe), there is no need to modify these values;

- `vocab_path_lang` and `vocab_path_lang`: A (relative) path to the pre-trained embeddings (vocabulary file and vectors file, respectively) of the language of the input texts, indicated by `texts_lang`. Only relevant if `texts_lang` is set to a value other than `en`.

Model Section. This section specifies the model to be used (type and a physical location) to segment texts. The two relevant configuration variables are as follows:

- `MODEL_TYPE`: specifies the type of the model – either a pre-trained model to be used to segment texts (if using `cats_predict.py`) or the model to be trained (if calling `cats_train.py`); There are two possible values for the model type: `cats` indicates the full-blown CATS model (without additional coherence modelling) and `tlt` is a weaker-performing segmentation-only model. For more details on both models, check the accompanying research paper;

- `MODEL_HOME`: specifies the path to the directory in which the pre-trained model can be found (if you're training the new model with `cats_train.py` then this is where the model will be stored). A pre-trained instance of the `cats` model is given in `data/models/cats_pretrained`, whereas an instance of the `tlt` model is available in `data/models/tlt_pretrained`. The values of `MODEL_TYPE` and `MODEL_HOME` must be aligned: if `MODEL_TYPE` is set to `cats` then `MODEL_HOME` must point to the directory where an instance of the `cats` model (not an instance of the `tlt` model!) is stored (i.e., `data/models/cats_pretrained`).

Architecture and Training Section. This section specifies parameters that are only relevant if you're aiming to train a new segmentation model instance (an instance of `cats` or `tlt`). For details on architecture and training (hyper)-parameters, check the Section 3.3 *Model Configuration* in the accompanying research paper.

4 Runnable Python Scripts

CATS contains three directly executable Python scripts. For the ease of usage (i.e., segmenting of texts with pre-trained models), we additionally couple the first two scripts (`cats_preprocess.py` and `cats_predict.py`) into a easy-to-run bash script `segment.sh` (see Section 2 `Quick Start`).

4.1 cats_preprocess.py

This script preprocesses the (possibly segmentation-annotated) texts and generates the corresponding data instances (TensorFlow records) that are being consumed by the segmentation models (either for training or for prediction). The script has the following arguments:

- `input_dir` specifies the directory that contains the raw texts (potentially annotated for segmentation; if we're creating the set of TF Records for training the segmentation model).
- `output_dir` specifies the directory in which the TensorFlow records, encoding the input texts, will be serialized; upon successful completion, two files will be created in the `output_dir`: `records.tf` and `blocks.pkl`.
- `--train`: this option (values `0` or `1`) indicates whether we are preprocessing the texts that contain (ground truth) segmentation annotations – to be used for training the segmentation models (value `1`) or we are only preprocessing texts (without the gold segmentation annotations) which we want to segment with a pre-trained model (value `0`). The default value is `0`. If `--train` is set to `1`, make sure that the string used to denote segment starts in the annotated textual files corresponds to the string value of `seg_start` in `config.py`.
- `--ssplit`: this option indicates whether the input texts are already sentence-segmented, that is, that each line of an input file corresponds to one sentence (value `1`) or not (raw text, not sentence-segmented; value `0`). If the value is set to `0` the content of each input file will first be sentence-segmented with the NLTK's sentence splitter. The non-segmented format is not allowed for preprocessing *training* files, since we expect to have segment annotations in separate lines (i.e., a combination `--train 1 --ssplit 0` is not allowed).

4.2 cats_predict.py

This script predicts segments using a pre-trained segmentation model (specified with `MODEL_HOME` and `MODEL_TYPE` in `config.py`). It uses as input the serialized files `records.tf` and `blocks.pkl`, previously generated from text files by `cats_preprocess.py`. The script `cats_predict.py` has these parameters:

- `input_dir` specifies the path to the directory containing the serialized input files `records.tf` and `blocks.pkl`, previously generated from text files using `cats_preprocess.py`;
- `output_dir` specifies the path to the directory where the segmented files will be stored. For each input text file there will be a corresponding segmented file created with an extension `.seg`;

- `--scores`: this option indicates whether a segmentation probability score (predicted by the segmentation model) should be printed next to each sentence in the segmented texts (for the pre-trained model, the segmentation probability thresholds are as follows: 0.3 for the `cats` instance in `data/models/cats_pretrained` and 0.5 for the `tlt` instance in `data/models/tlt_pretrained`).

4.3 cats_train.py

This script trains the new segmentation model from scratch (an instance of the `cats` or an instance of the `tlt` model, depending on the value of `MODEL_TYPE` in `config.py`). The model is trained on the `records.tf` file, previously created from a segment-annotated training set of textual documents, using the `cats_preprocess.py` script with the flag `--train` set to `1`. The model that is trained will be stored in the directory specified by `MODEL_TYPE` in `config.py`. The script `cats_train.py` has no arguments – the path to the file (`records.tf`) containing the serialized training set (collection of TensorFlow records) needs to be set in the `config.py`, as the value of the parameter `tfrec_train`.

5 Dependencies and Hardware Requirements

CATS tools have the following prominent Python library dependencies:

- `Tensorflow` (tested with version 1.12)
- `NLTK` (tested with version 3.4)
- `Numpy` (tested with version 1.15.4)

For the training of the models, it is recommended to have access to Graphical Processing Units (GPUs) and the GPU version of Tensorflow installed. If the CATS tool is to be used only the segment texts using the provided pre-trained models, this is feasible on CPUs as well (albeit it is going to be slower than running on GPUs by a factor of 2-3). Running the pre-trained models to make segmentation prediction requires ca. 4GB working memory (RAM). Training the models from scratch requires 12GB working memory (RAM).