**A Quick Start for the Zero-Example Video Search Tool**

## There are mainly two programs in the project package

* The main program *avs\_console\_main*: used for everything except the processing between Step 1 and 2.
* A python script *query-to-concept\_mapping*: used for calculating the word-to-word distance matrix by WordNet between Step 1 and 2.

## Installation

* C# runtime (Visual Studio for Windows or Mono for Linux) with .NET framework 4.5.2. To run the executable on Windows without compilation, you may install .NET framework only.
* Python 2.7 and [NLTK package](http://www.nltk.org/install.html) with [WordNet configured](http://www.nltk.org/data.html).

## Set-up

* Simple! Just fill in the required paths in *config.ini* in the main program package.
* The following paths are required:
  + *queryDescriptionFile = "XXXXXX" // your query goes here*
  + *conceptListFile = " XXXXXX"*
  + *idfTableFile = "XXXXXX"*
  + *similarityTableFile = "XXXXXX" // this file is to be generated by the python script, the path must match the output in the python script*
  + *mergedInfo\_nConceptsFile = "XXXXXX"*
  + *conceptResp\_TableFile = "XXXXXX"*
  + *conceptResp\_VideoListFile = "XXXXXX"*
  + *mappingResultDir = "XXXXXX" // this is where all the results should go*
* Then, edit the paths in the python script, point them to the corresponding files:
  + *InputFile = r'XXXXXX' // must point to “$mappingResultDir$\Event\_Concept\_TermTable.txt”*
  + *OutputFile = r'XXXXXX' // copy this path to similarityTableFile above*
* Done.
* Being lazy? We have both binaries and sample *config.ini* files already set up for you. Just check the *config.XXX.ini* files, each corresponding to one dataset. Choose one and rename it to *config.ini*, correct the paths in the python script per the file you have chosen, then you are good to go.

## How to play

* Use the following console commands:
  + --exit: A command everyone likes.
  + --loadscoretable: Load the concept response table for a video/frame corpus.
  + --procstep {1-4}: For details, please read the comments on *processingStep* in *config.ini*.
* A dry run:
  + >>> --loadscoretable
  + >>> --procstep 1
  + *(You should find a file named Event\_Concept\_TermTable.txt generated in your $mappingResultDir$.)*
  + Run the python script until it says “Job done.”
  + *(You should find the similarity table file generated in your $mappingResultDir$.)*
  + >>> --procstep 2
  + *(You should find the concept candidates, a.k.a. semantic query, generated in a folder.)*
  + >>> --procstep 3
  + *(You should find the video/shot ranking results generated in a folder.)*
  + *Alternatively, you can use “--procstep 4” to combine 2 and 3 in a single command. Processing step 2 will print out the ranked concepts for each query. This is used for concept screening. Step 3 alone will NOT print the concepts. Note that “--loadscoretable” is not required for step 2.*

## Interactive search

* In case a user modifies a query in *$queryDescriptionFile$* during runtime, do the following:
  + >>> --reloadquery
  + *(If there are new words added to the query, the following two steps are required, otherwise just jump to “--procstep {2-4}.” Note that new words mean the words unseen in the vocabulary of the old query.)*
  + >>> --procstep 1 …………………………………………………………………… *(if there are new words)*
  + Run the python script again until it says “Job done.” …….…………...... *(if there are new words)*
  + *(CAUTION: You may encounter a runtime error if the vocabulary is updated but the above two steps are omitted!)*
  + >>> --procstep 4
  + *(This will update all the results.)*
* Concept screening:
  + *To enable, you need to set “isUseHandPickedConcepts” true and set a valid path on the line with “conceptHandPickedPerQueryFile” in config.ini.*
  + *After processing step 2, inspect the ranked concepts in $mappingResultDir$\RankedConcepts for each query. Pick up the relevant and discriminative concepts, and write them down into conceptHandPickedPerQueryFile for the corresponding query.*
  + *You can refer to datasets\AVS16\IACC.3\_SRIP2K\_DRN\_Handpick as an example. Each row is a selected concept. The concepts that are not written in the hand-pick file are dropped out. The format in each row is {concept\_id}[\t{reranking\_order}][\t{override\_weight}]. The latter two are optional. Use “-1” as a placeholder. This can be useful if you want to use {override\_weight} without {reranking\_order}. If, for a query, the hand-pick file is not found, the automatic concept ranking will be used instead for this query.*
  + >>> --procstep 3
  + *(This will update video/shot ranking result.)*
  + *Alternative to hand-pick concepts, you can use a black list to mask unwanted concepts. The black list uses the name of the concept instead of the concept ID. The name must be the same as it appears in the conceptname\_list.txt file in the features folder. Refer to datasets\MED14Test\SFRISP\_DRN\_2774\_Blacklist for an example.*

## TIPS

* Comment a line in any configuration file by placing a “;” at the beginning of the line.
* The main program is capable of simple pre-processing like *stop word removal* and *lemmatization*. As a result, simple queries can be directly used as input without NLP parsing beforehand. In this case, refer to datasets\AVS16\Queries\avs16\_queries\_auto.txt.
* You can skip a stop word by placing a “+” right in front of a word in the query. For example, “+person”.
* A user can iteratively do the concept screening or query modification, and inspect the search result.
* Black lists can be used collaboratively with the hand-pick concepts.
* “--procstep 1” and “--procstep 2” can be used without “--loadscoretable” command.
* nTopConceptsOnly in *config.ini* is useful to cut off to the top-N concepts selected for each query. This usually gives a better performance. To disable the cut-off, set this option to -1.
* Use some tiny code in tools folder for your own dataset and features.
* Please cite our work if you find this tool helpful ☺