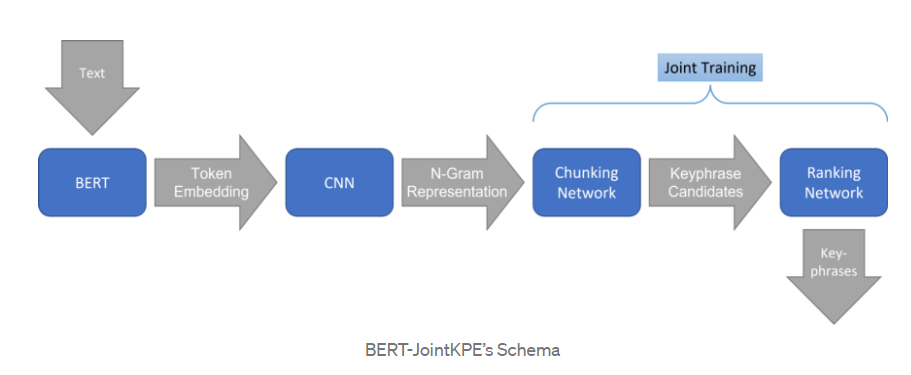
## Week 1 07/02 - 11/02

* Get started at sinequa
* See video of introduction of sinequa
* Install environment (pycharm, rider, git)
* Read previous work + KPE sinequa medium
* Ill-defined task : vast number of potentially relevant KP + dont have definition of good KP -> use F1@5 score : F1 score on exact match of the top 5 extracted KP
* Scarcity of data : 39 datasets , scientific papers, 25 datasets in english, -> need bigger dataset
* Using of HAL dataset : article + set of KP
* Model choice :
  + Unsupervised learning is not good in term of unsatisfactory kp and robustness
  + Supervised learning :
* Meta learning : learning to learning , model learns the general principle of what is a good KP -> zero shot performance is expected to rise. Using parameters : annotator, structure of the documents, topic of the document and language
  + The more diverse the added data, the more this effect is noticeable. But, using 5 is barely better than using 2
    - Larger the dataset, more useful meta-learning
    - Using dataset from a certain domain improve zero-shot performance on another dataset with same domain
    - Small datasets can make a real difference
    - Combine small ds with large ds
    - Repeat small ds to size of big ds
* Influence of model size : reduce weight but not reduce the performance that much
* Post processing tricks
  + Only keep kp with high confidence score
  + Syntactic diversity : delete shorter KP if it is included in another KP
  + Merge overlapping KP :
  + Remove stop word of begin or the end of KP

### 08/02/2022

* Read previous work and other project
* Run code python of previous work
* See video about git
* Ask about project with Boris
* Install Python environment
* Continue seeing video

### 10/02/2022

* Understanding implementation of JointKPE
* Dataloader.build\_dataset()
* train(args, data\_loader, model, train\_input\_refactor, stats, writer)
  + args
  + Data\_loader
  + Model : implementation of model with forward() -> return loss function to have a customized loss function
  + Train\_input\_refactor
  + Stats
  + Writer

### 11/02/2022

* ~~Ask boris~~
* Run code test code, not train + small dataset to test on
* Project would be to deliver a model that can run the inference in production ? + what is ONNX model
* Rewrite code in tensorflow ? + what part to write in C#
* Using of metadata
* ~~Plan next week~~
* See feature of raw, preprocessing and feature cached data -> find way to improve
* Read and compare preprocess of JointKPE
* Split small piece of data to train and test
* Reproduce last work by running train test on data (what data : on each small dataset or combine to a big one)
* Read Antoine report
* Read new paper about KPE if have time
* Pass sinequa university

## Week 2 14/02 - 19/02

* See feature of raw, preprocessing and feature cached data -> find way to improve

Track to improve :

* Reproduce the result of Antoine
* Change the length of keyphrase (why it should be 5, can it be different, if yes, what should it is -> find paper)
* Analyze on different datasets (new dataset or old dataset but see new features in it, see Steffen’s exemple)
* Use different length tokens of document ( speed up the train)
* Clean candidate keyphrase to see if there are improvements
* Span propose from QUaAS project

Question

* Transformation from txt to json
* Why dont using openkp and kp20k
* Read and compare preprocess of JointKPE
* ~~Split small piece of data to train and test~~
* ~~Reproduce last work by running train test on data (what data : on each small dataset or combine to a big one)~~
* ~~Read Antoine report~~
* Read new paper about KPE if have time
* How to evaluate a model : high f1@5 score ? why not different. Current model have impressive result only on Inspec, is it a good think to follow
* Phraseformer
* Need to read more about prosperity of each dataset
* ~~Plan next week~~
* Find check point of Antoine to evaluate
* Construct a clean repo github
* Construct a data analyze table of of all dataset
* Write code preprocessing of extract only good phrase, and find a way to test if this idea is interesting
* Loss function to evaluate KPE model -> read paper
* Construct a plan of training
* Test MLzilla

### 14/02/2022

* ~~See feature of raw, preprocessing and feature cached data~~
* In raw data of kp20k : abstract (string), keyword(list string), title -> prepro : url(int) doc\_words(list of string), keyphrases(list of list string -> each list string is a keyphrase), start\_en\_pos (list of interval, an interval is 2 number represents the begin and end position of KP) -> cached feature
* In raw data of openkp : url (string - link to documents) text(string - documents) VDOM (list of dictionnaty (id : index, text : keypharase, feature - \*not knowing\*, start\_idx + end\_idx : position)) -> prepro : url (string - link to documents), doc\_words (list string - documents) word2block (segment of a phrase) bloc\_features(list of list float - *not knowing*) keyphrases (list of list string, each list string is keypharas) start\_end\_pos (list of interval), present\_keypharses (list of list string - *not knowing*) -> cached feature
* Cached feature is construred from data preprocessing, using from prerpo : url, doc\_words, keyphrases, start\_end\_pos. Feature of cached feature :
  + url -> get from prepro url
  + tokens -> get from tokenize\_output tokens
  + Valide\_mask -> get from tokenize\_output valid\_mask
  + Doc\_words get from prepro doc words
  + phrase\_list -> get from info\_or\_label(tot\_pharase\_list) : all phrases possible of the document, limit by max\_phrase\_length
  + mention \_lists -> get from info\_or\_label(tot\_mentions\_list), list of list of int, each list of int is the position where the phrase with the length i is first mentioned -> to train pooling the same phrase
  + Keyphrases -> get from prepro
  + ngram\_label -> get from info\_to\_label(ngram\_label) : similarity between a phrase to all other phrase
  + chunk\_label -> get from info\_to\_label(chunk\_label) :
    - Info\_to\_label using filter positions (true label), tot\_mention\_list(list index of all possible phrases) and differ\_phrase\_num(number of all phrase possible)
* What feature of cached feature is used to train ?
  + tokens
  + valid mask
  + mentions\_lists
  + ngram\_label (if train)
  + chunk\_label (if train)
  + phrase\_list (if eval or test)

### 15/02/2022

### 16/02/2022

* ~~Split small piece of data to train and test~~
* Finish on boarding cours Sinequa university
* Fix train function of Antoine code

## Week 3 21/02 - 25/02

* ~~Find check point of Antoine to evaluate (no need any more)~~
* ~~Construct a clean repo github (wait for boris and loic)~~
* ~~Construct a data analyze table of of all dataset~~
* Can change the max length kp 5 -> 4 because most of the kp is length 1 - 3
* Lemminization can also be a choice
* Each dataset has a very different distribution of length of document, should we make a pivot to have the same length -> question token length and document length
* Write code preprocessing of extract only good phrase, and find a way to test if this idea is interesting
* Data Inspec : there is keyword with punctuation -> should we clean the punctuation
* Keyphrase does not start or end with a stopwords, we can also clear it in preprocess
* ~~Loss function to evaluate KPE model -> read paper~~
* f1@5 score : for each do,get top 5 kp of candidate, check if it is in list of reference, if yes, true positive + 1, -> calculate precision(true positive / 5), recall (true positive / len(reference) on each document -> calculate f1 score on each document. f1@5 score of a dataset is mean f1@5 score on all doc of dataset
* ~~Test MLzilla~~

Work after boris meeting

* ~~Fix bug of log writing~~
* ~~Fix bug tensor board installation~~

(still dont understand why -> need to check for each time create new envs)

* ~~See repo preparation of ranker~~
* ‘Must have’ directory or file :
  + ‘Name of project’ : code of training process, preprocess, model, metric, technique to train
  + Resources : result, charts of result, track of used dataset, track of best performance
  + Scripts : all file that can run
  + Evaluation
  + Process\_description : files md to track what we do
  + Readme
  + Requirements.txt
  + setup.py
* What to go where in my repo
  + Requirements.txt setup.py outside
  + KPE : all code that used for 2 preprocess data, bert2joint, etc …
  + Resource : analyze, result, train log, test log, track of used dataset
  + Processing descriptions
  + Scripts : download and preprocessing data, train, test, data clean, data augmentation
* ~~Check on dvc -> how to manage used dataset (way for github repo task)~~
* <https://github.com/iterative/dvc#comparison-to-related-technologies>
* https://dvc.org/doc/start
* ~~Check on how to split train test dev, reproduce it (pubmed and semeval 17)~~
* ~~Connect to mlzilla (ask cyril)~~
* ~~Reproduce on mlzilla~~
  + <https://pytorch.org/> - should install pytorch of environment by this link -> adapt to cuda driver version
  + Need tensorboard 2.8.0 ->
  + It take only ⅓ of GPU, we can increase the batch size to 2 times
* ~~Extra task : read bert tag kpe and span kpe~~
* Extra task : read diff roberta vs bert vs distill bert

classifier\_scores = classifier\_scores.masked\_fill(

mask=active\_mask, value=-float("inf")) -> why have this

## Resume of work with boris

* ~~Reproducing BERT-KPE results on openkp - ok (with bigger kp > 5)~~
* ~~Reproducing BERT-KPE results on openkp20k - ? (kp up to 5)~~

### Discuss the data that we will use in production

* ~~- Recover the (5 dataset) original data - ok~~
* - Have a good way to present the statistics
  + - Jupyter notebooks results to wiki page on the repo ?
* - See if there are results on other papers about these datasets - ?
  + - Is it split the same way ?
  + - Is the evaluation metric the same ? Example: f1-score, f1-score@?...
  + - Converge on a metric that we can use maybe to compare to other papers
* - Converge to a pre-processing way to use these dataset and potential format Example is Antoine pivot format good enough ?
* Good point
  + Same format for preprocessing data
* Week point
  + Name of file is a mess,
  + Too many functions with same functionality
  + Overlen document is treated with cut only
  + Not delete yet hidden keyphrase, should delete also doc with no kp
* ~~- Find a way to store the pivot dataset (jsonl for example) using dvc~~
  + ~~- Location drive, repo, google storage bucket ?~~

### Training and evaluation of the model

* - Discuss which data we want to use to train our model that will be used in production
  + - Train and eval on each of the "free" dataset
  + - Train on one dataset and evaluate on openkp or kp20k test sets (Zero shot)
  + - Think about a concatenation method to improve the results on the different test set
* ~~- Keep track of all the trainings using mlflow (ask Boris)~~
  + ~~- Plug mlflow to torch training/code - ?~~
* - Update the wiki benchmark of your repo regularly - ?
* - Try to convert a pythorch model to Onnx - ?
  + - Have a look at the MachineLeanring/OnnxConversionLab repo - ?

<https://pytorch.org/tutorials/advanced/super_resolution_with_onnxruntime.html>

<https://pytorch.org/docs/master/onnx.html>

<https://github.sinequa.com/MachineLearning/OnnxConversionLab>

### Production integration

* - Have working Sinequa dev env - ?
* - Discuss with Emilie and Loïc about the specs... - ?
* - Have a look at other model configurations in Sinequa - ?
* - Start the integration

## Week 4 28/02 - 04/03

* ~~- Keep track of all the trainings using mlflow (ask Boris)~~
  + ~~- Plug mlflow to torch training/code - ?~~
* ~~Reproducing BERT-KPE results on openkp20k - ? (kp up to 5)~~
* - Converge on a pre-processing way to use these dataset and potential format - ? Example is Antoine pivot format good enough ?
* Good point
  + Same format for preprocessing data
* Week point
  + Name of file is a mess,
  + Too many functions with same functionality
  + Overlen document is treated with cut only
  + Not delete yet hidden keyphrase, should delete also doc with no kp
* -> finish split long document
* ~~- Have working Sinequa dev env - ?~~
* Use devinst to install, build and update
  + devinst --ice-path=ice update
  + devinst --ice-path=ice compile
  + In distri/website/bin, run sinequa.admin.exe -> start service before run the application
* ask about what to do in dev env
* Ask about SVN
* ~~Discus with boris~~
* Ask about mlflow -
  + Log model and predict by this model
  + List parametre concerned
  + how to install docker
  + run on google cloud
* Prepare for repo github
  + The proposed structure
    - ‘Name of project’ :
      * code of model
      * training process
    - Data : dvc file (to clone data from google drive or a repo) -> after clone, it would have : dataset, prepro\_data, cached\_dir, preprocess.python + preprocess.sh
    - Resources : result + mlruns (mlflow)-> use wiki
    - Scripts : all file that can run
    - Evaluation
      * Test process -> maybe not need
    - Process\_description : files md to track what we do
    - Readme.md
    - Requirements.txt
    - setup.py
  + Only bert joint + roberta or all -> only code of bertjoint
  + The code of mlflow in it -> have code of ml but ml runs should be on the main app
  + The folder for result, mlruns and dataset -> use wiki
* For dataset
  + Present the problem of Antoine pivot
  + Solution for long document
  + Is an MIT license possible to use ?
  + What use case of Sinequa
* For SOTA
  + How to calculate the f1 score in jointKPE context

-> change code see the result on data mlflow main app - done

-> create the repo as we want

-> make the function for preprocess of every version possible

->

-> wait boris for MIT license, F1 score,

### List bug:

* Change max phrase length
* ~~Dvc on mlzilla~~

## TODO list for next time

* Run on all dataset and test on zero shot
* ~~Research on license of dataset~~
* ~~Calculate inference time as a metric~~
* ~~Preprocess data to concatenate dataset -> to check if there is improvement~~
  + ~~Script to concatenate~~
  + ~~check~~
* ~~Refactor code to not depend on name data set~~
* ~~Refactor preprocess; only 2 code : convert code to pivot, and pivot to input~~
  + ~~Raw to convert~~
  + ~~Convert to pivot~~
  + ~~Pivot to input~~
* ~~Pytorch lightning~~
  + ~~Test on colab~~
  + ~~Find way to use TPU of sinequa~~
  + ~~Re-implement JointKPE in lightning~~
* ~~Read smart bert paper to understand if it is worthy to test~~
* ~~Fix bug batch size~~
* ~~Run~~ [~~smart bert~~](https://github.com/victorywys/SMART-KPE)~~, test on all dataset~~
* ~~Read paper of SOTA~~ 
  + ~~Check their metric~~
  + re -implement their metric in joint kpe
  + Find code or implement
* Script to change from checkpoint model pytorch to ONNX + script to import ONNX model and make prediction
* Test train by remove phrase with punctuation and stop words
* Few shot learning
* Write wiki page or repo
* - Have a look at other model configurations in Sinequa - ?
  + Code in helium.ice/sinequa/ml/Classification/Api/ClassificationApi.cs
  + Code in helium.ice/sinequa/Configuration/CCTfClassification.cs
  + <https://doc.sinequa.com/en.sinequa-es.v11/Content/en.sinequa-es.admin-ui-query-intents-v2.html>
  + C:\Dev\ice\distrib\forms / C:\Dev\ice\distrib\data\configuration\tfclassifications /

## Week 5 07/03 - 11/03

* ~~- Try to convert a pythorch model to Onnx - ?~~ 
  + ~~- Ask Remi how to know it successfully~~
  + ~~- Have a look at the MachineLeanring/OnnxConversionLab repo - ?~~
* ~~Refactor code to not depend on name data set~~
* ~~Refactor preprocess; only 2 code : convert code to pivot, and pivot to input~~
  + ~~Raw to convert~~
  + ~~Convert to pivot~~
  + ~~Pivot to input~~
* ~~Calculate inference time as a metric~~
* ~~Preprocess data to concatenate dataset -> to check if there is improvement~~
  + ~~Script to concatenate~~
  + ~~Check~~
* Implement shuffle dataset

## Week 6 14/03 - 18/03

* Pytorch lightning and debug on GPU
* Implement feature cached dir
* ~~Remove stop word~~

## Week 7 21/03 - 25/03

* Pytorch lightning and debug on colab
  + Iterable distributed dataset
  + Refactor code
* Improve tokenizer
* ~~Read paper~~
* ~~Read Classification configuration~~
* ~~- Have a look at other model configurations in Sinequa - ?~~
  + ~~Code in helium.ice/sinequa/ml/Classification/Api/ClassificationApi.cs~~
* Important functionalities:
  + Load model + load model for tester
  + Unload model
  + Serve : serve a model for a deployment given the deployment name
  + Most of function are called from ClassificationApiV1
* How the classification train the model, is it the same on KPE
  + Code in helium.ice/sinequa/Configuration/CCTfClassification.cs
  + <https://doc.sinequa.com/en.sinequa-es.v11/Content/en.sinequa-es.admin-ui-query-intents-v2.html>
  + C:\Dev\ice\distrib\forms / C:\Dev\ice\distrib\data\configuration\tfclassifications /

## Week 8 28/03 -01/04

Reunion with boris

1. Data pipeline

* From git to json file to preprocess to input data
* Why need input data

1. Implementation of pytorch lightning

* Use of override function of lightning
* Some function underhood
* Should we get rid of customized transformer
* Len of dataset by a config file
* Distributed iterable dataset
* Out of memory error and need of colab pro

1. Another thing

* Problem of tokenizer
* Problem of length of document
* Benchmark
* Smart bert can just be used with VDOM
* Candidates :
  + TNT-KID: Transformer-based neural tagger for keyword identification
  + UCPhrase: Unsupervised Context-aware Quality Phrase Tagging
* Question for configuration of KPE

## Week 9 04/04 -08/04

* Presentation
* Test with different DLM
  + Result on distil bert is still good
  + -> maybe the reason is cased and uncased
  + -> change to bert and distilbert -> no more roberta
* Debug colab lightning

## Week 10 11/04 -15/04

* Analyse result, update wiki benchmark
* Onnx conversion
  + Still have error of batch size
* Unit test for post processing and preprocessing python side
* Unit test for post processing and preprocessing C# side -> first try

## Week 11 18/04 -22/04

* Preprocessing and post processing for long document
* Cleaning the pull request
  + Remove all code unused
  + Comment for maintenance.
* Unit test for post processing and preprocessing C# side
  + Remain the preprocessing of Remi for batcher
  + -> Rewrite KeFeature
* Launch training on kptimes splitter and combined for dataset with kptimes splitter
  + -> result on splitter improve a little the performance
  + -> model of only kptimes splitter is not quite good on zero shot learning, but just a very small portion of another dataset can improve dramatically it

## Week 12 25/04 -29/04

* Evaluate post processing for long document
  + Analyze the result on pubmed and kptimes
  + Import and preprocess ldpk10k
* Implement build input feature in C#
* -> train on a small portion of data is enough
* Launch training on kptimes position and combined for dataset with kptimes position
  + -> result

## Week 13 02/05 -06/05

* Evaluate post processing for long document
* Debug colab
* Read paper -> result in analysis of long document

## Week 14 09/05 -13/05

* Unit test for long document processing in python side
* post processing to construct list of different ngam from original test
* post processing to have clean results ( not repeated ngam)
  + Stemmization
  + Repeat by containing an apart
* Encapsulate the function of creating active Mask
* try serving onnx in scmd

## Week 15 16/05 -20/05

* Implement KeywordExtractionModel + unit tests:
  + Load model
    - Using DL model, implement the class KeywordExtractionConfiguration to contain the configuration of the model
  + Unload model
    - Using dispose function
  + Serve mode
    - Using KeywordExtractionHelper to preprocess the text
    - Using KeywordExtractionBatcher to create batch of data
    - Using DlData to contain the feature, using KeywordExtractionFeature to write input feature from text
    - Serve model
    - Using KeywordExtractionHelper to extract the keyphrase and give the result -> implement the class KeywordExtractionResult represents the result of the model
* Implement the KeywordExtractionApi + unit test (using KeywordExtractionModel)
* Implement Serving routes
* Testing Api with PostMan
  + Compare and fix bug, to have same result on python side
* On research side
  + Implement the metric to calculate the inference time on each doc
  + Testing model with reduce the max ngram length (from 5 -> 4): the result reduces 1%.
  + Testing Inference time on doc

## Week 16 23/05 -25/05

* Implement KeywordExtractionModelManager:
  + Using a model store which map a model name to a instance of KeywordExtractionModel
  + Implement in KeywordExtractionConfiguration to transform from model name to configuration and reverse
  + Implement load, serve and unload of KeywordExtractionModelManager using model name
* Adapt model manager to api, change Api so it can run using a model name
* Implement Post processing for long document:
  + For one long document, the batcher can batch into some instances of data, the post processing aggregate the result to have only on result

## Week 17 30/05 -03/06

* Clean code
* On python side: one script to run from pivot format to train model -> have one configuration
* Implement Loading Routes and Unloading routes
* Read Command Sinequa
* Research part:
  + Spanbert
  + Mean in total score

## Week 18 07/06 -10/06

* Clean code python
* Implement keyword extraction command
  + Implement class KeywordExtractionPrediction
    - Using sqlConfig to construct a SQL
    - Using a CursorHelper to read result of sql query
    - Read from configuration and command line to construct configuration of keyword extraction
    - Using index updater to column of prediction
  + Implement command type keyword extraction
  + Define configuration
* Run experiment with change warming up step = 10%
  + Improve score

## Week 19 13/06 -17/06

* Implement Command Text Augmentation
  + Possibility to add whatever a runnable model from text augmentation folder
* Clean code python -> refactor function of evaluation
* Documentation of Model Delivery
* Use BenchPress to benchmark speed of onnx model
* Read paper CogLtx -> apply for long document
* Experiment model with bert small, bert medium, bert mini, bert tiny

## Week 20 20/06 -24/06

## Week 21 27/06 -01/07

## Week 22 04/07-