Log file analysis

Overview

- Goal: predict the operator action, requested memory, job splitting, xrootd enabled

- So far: each feature is a sparse matrix (error/site)

- Entries of the matrix is the number of times the error code is thrown per site

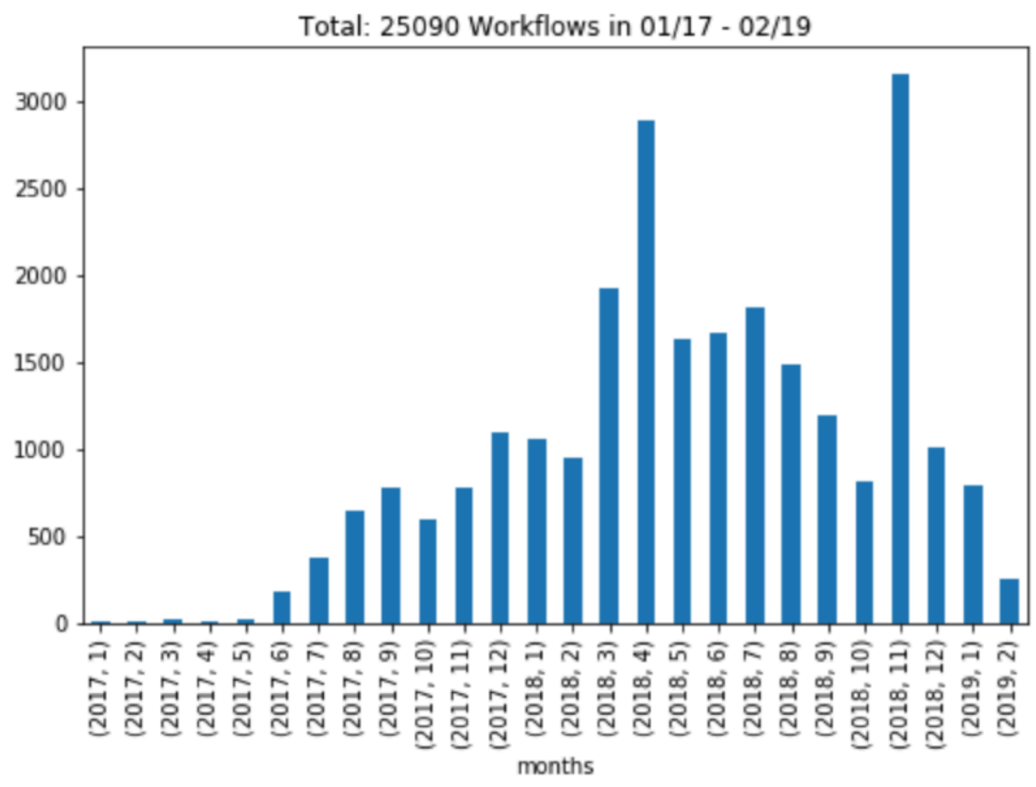
- Idea: add NLP info from log files to provide more information

	site 1	site 2	site 3	site 4	. . .	site 140	site 141	site 142
error 1	1	1	1	1	1	1	1	1	1	1
error 2	4	0	0	0	0	0	0	0	0	0
error 3	8	1	0	0	0	0	0	0	0	0
. . .	0	2	0	0	0	0	0	0	2	0
error 52	0	0	0	0	0	0	0	0	0	5
error 53	0	0	0	0	0	0	0	0	0	1
error 54	1	0	0	0	0	0	0	0	0	0

Error matrix (taken from Dominykas)

NLP: Acquisition of text snippets

- ~ 25.000 workflows
- Small error messages stored on HDFS with WMArchive
- Full logs partially on CASTOR / EOS
- Use pyspark for reduction and write back result to HDFS



Workflows per month from actionshistory.json

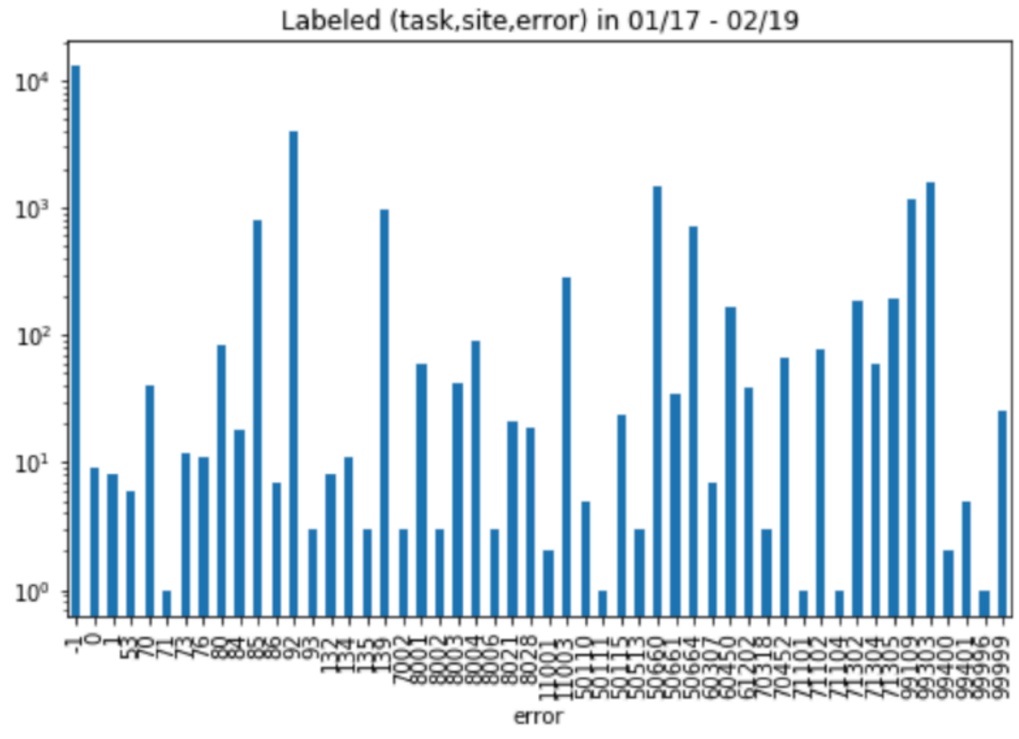
1. Small messages from WMArchive

- SWAN notebook for analysis - faster than WMArchive framework

- Interesting key to identify log file: (task_name, site, error)

- -1 error codes cannot be identified

- Runtime ~ 20 min



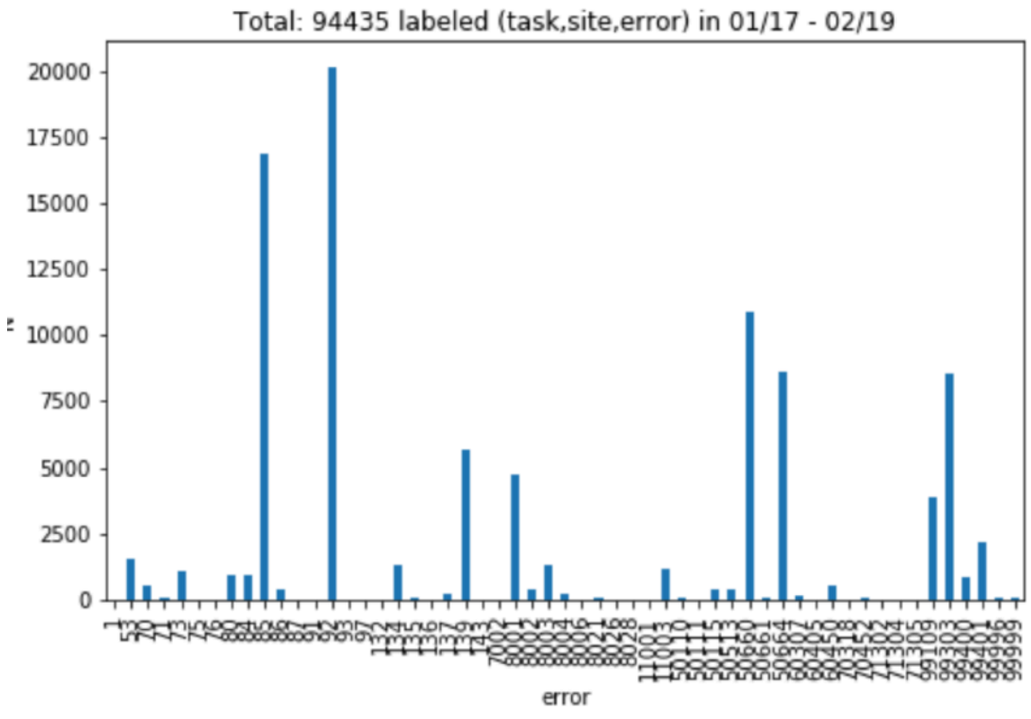
Error codes for (task, site, error)

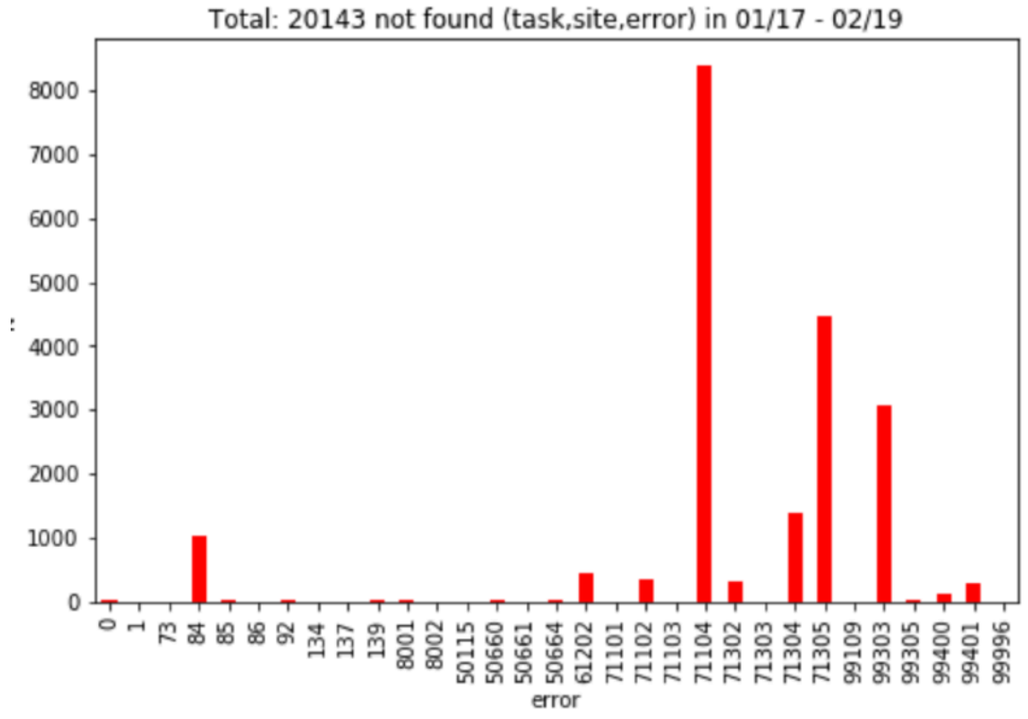
WMarchive entry example:

1. Select failed workflows
2. Flatmap the entry to key, value: (task, site, error), msg
3. Join on this key with the actionshist entries for the labels


```
{ …,
u'task': u’/vlimant_ACDC0_task_HIG-RunIIFall17wmLHEGS-01415__v1_T_180706_002124_986/HIG-
RunIIFall17DRPremix-02001_1/HIG-RunIIFall17DRPremix-02001_1MergeAODSIMoutput/HIG-
RunIIFall17MiniAODv2-01299_0', … ,

u'meta_data': {u'jobstate': u’jobfailed'}

u'steps': [

    {u'status': 99996, u'errors': [
        {u'type': u'ReportManipulatingError', u'details': u'Failed to find a step report for
         stageOut1!’,
         u'exitCode': 99996}], u'name': u'stageOut1', u'stop': None, u'site': u'T1_UK_RAL', },

    {u'status': 0, u'errors': [], u'name': u’logArch1’},

    {u'status': 85, u'errors': [

          {u'type': u'CMSSWStepFailure', u'details': u"\n Adding last 25 lines of CMSSW stdout: ',
           u'exitCode': 85},

          {u'type': u'Fatal Exception', u'details': u"An exception of category 'FileReadError'
           occurred while\n    ', u'exitCode': 8021},

          {u'type': u'ErrorLoggingAddition', u'details': u'Adding extra error in order to hold error
          report\n\nAdding last ten lines of CMSSW stderr:\nWARNING:',  u'exitCode': 99999},

          {u'type': u'WMAgentStepExecutionError', u'details': u"<@========== WMException Start
          ==========@>\nException Class: CmsRunFailure\nMessage: ", u'exitCode': 85}],
     u'site': u'T1_UK_RAL', }], …
}
```

Results





~ 95.000 correctly associated keys with at least one error message

~ 20.000 messages not found

- Some entries (error != -1) are not found (e.g. site field not filled)
- What is the meaning of status?
- Other fields interesting?
- Multiple error messages per key redundant?

2. Full logs from EOS / CASTOR

- Full logs contain more information
- Goal: get more and larger snippets than what is stored on WMArchive ~ more lines around error
- Problem: log archives are collected in large log collect tar files
- Used WMArchive to associate lfn of logArchive with lfn of logCollect to get the path on EOS/ CASTOR
- move also to notebooks

```
../../_condor_stderr
../../_condor_stdout
../../wmagentJob.log
../../job/WMTaskSpace/cmsRun1/cmsRun1-stderr.log
../../job/WMTaskSpace/cmsRun1/FrameworkJobReport.xml
../../job/WMTaskSpace/cmsRun1/cmsRun1-stdout.log
../../job/WMTaskSpace/cmsRun1/scramOutput.log
../../job/WMTaskSpace/cmsRun1/Report.pkl
../../job/WMTaskSpace/cmsRun1/PSet.py
../../job/WMTaskSpace/cmsRun1/PSet.pkl
```

Content of log archive tar.gz

Pipeline from CASTOR to HDFS

- Problem 1: data on CASTOR difficult to access -> only certain CMS accounts have rights to stage

- Problem 2: need to stage many large tar files ~PB

- Problem 3: Staging takes some time: CMS experts estimated 1-2 months, but IT experts 1-2 weeks

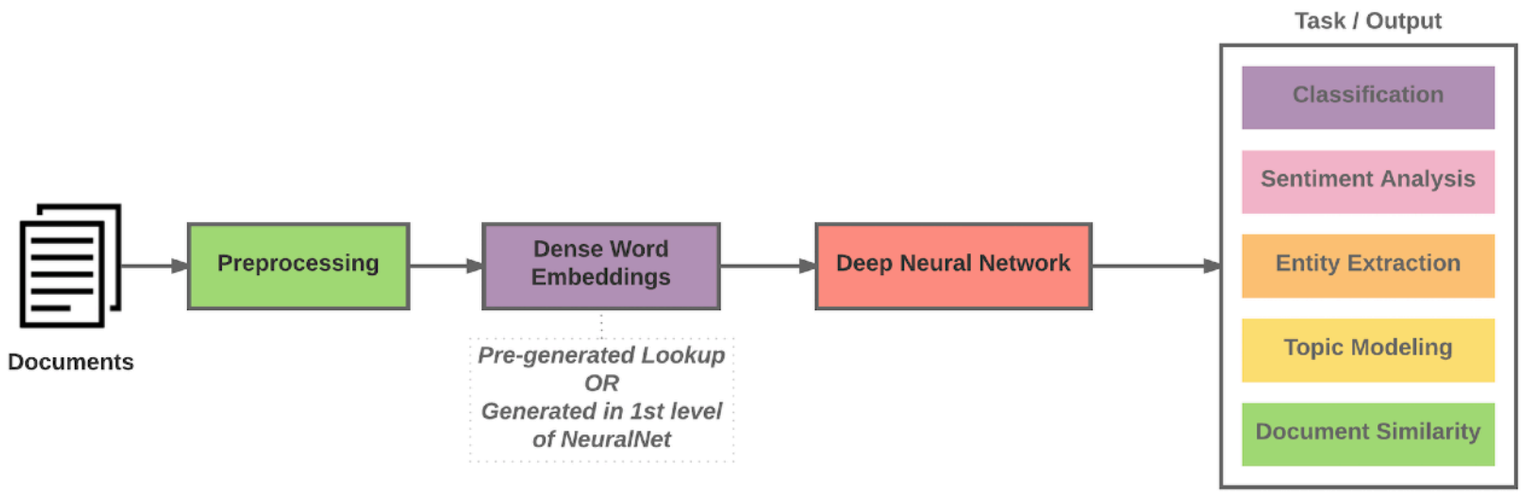
- Help from CERN IT: notebook to stream large tar logCollect files directly from EOS to access content of small tar.gz logArchives with pyspark and store relevant infos to HDFS -> verified with small number of files

- Our proposal: first show improvement with small error snippets

NLP with the reduced error messages

- Depending on the size of the output the word embeddings and the ML have to be done with pyspark

- Considering only one error message per (task_name, site, error) -> NLP can be done with pandas



Preprocessing

- Challenge: Logs are not human language
- Tokenize + remove unnecessary words
- Also more sophisticated clustering algorithms available: https://github.com/logpai/logparser
- But usable with spark?

```
Sep 19 19:18:40 acmepayroll syslog: 04/30/10
12:18:51 39480627 wksh: HANDLING TELNET CALL
(User: root, Branch: ABCDE, Client: 10101)
pid=9644
```

WhitespaceTokenizer

1. Sep | 19 | 19:18:40 | acmepayroll | syslog: | 04/30/10 | 12:18:51 | 39480627 | wksh: | HANDLING | TELNET | CALL | (User: | root, | Branch: | ABCDE, | Client: | 10101) | pid=9644

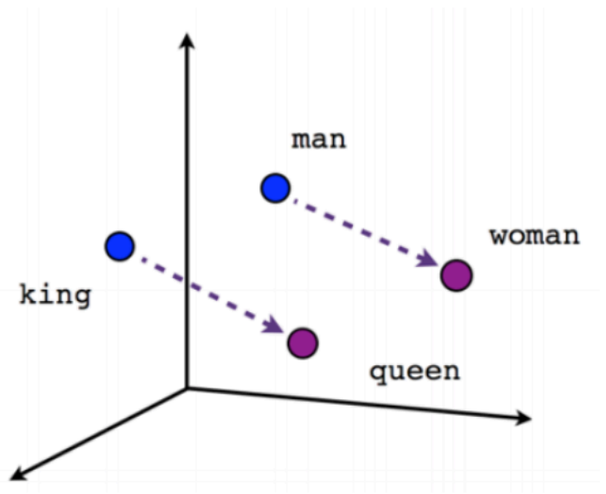
WordPunctTokenizer

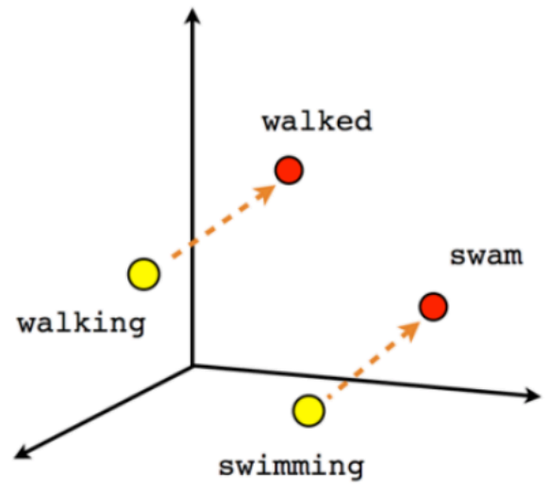
1. Sep | 19 | 19 | : | 18 | : | 40 | acmepayroll | syslog | : | 04 | / | 30 | / | 10 | 12 | : | 18 | : | 51 | 39480627 | wksh | : | HANDLING | TELNET | CALL | (| User | : | root | , | Branch | : | ABCDE | , | Client | :

TreebankWordTokenizer

1. Sep | 19 | 19:18:40 | acmepayroll | syslog | : | 04/30/10 | 12:18:51 | 39480627 | wksh | : | HANDLING | TELNET | CALL | (| User | : | root | , | Branch | : | ABCDE | , | Client | : | 10101 |) | pid=9644

Word Embeddings





- Vector space models: map words in a continuous vector space where semantically similar words are mapped to nearby points

- word2vec: feed a corpus -> get one vector for each word

- Default word2vec on spark: average all word vectors per document -> one high dimensional vector represents document

- Also interesting: counting approaches TF-IDF, pre-tained models

Example pipeline: unique error message per (task, site, error)
Tokenization + word2vec with pyspark
Join + group word vectors and labeled history on task_name

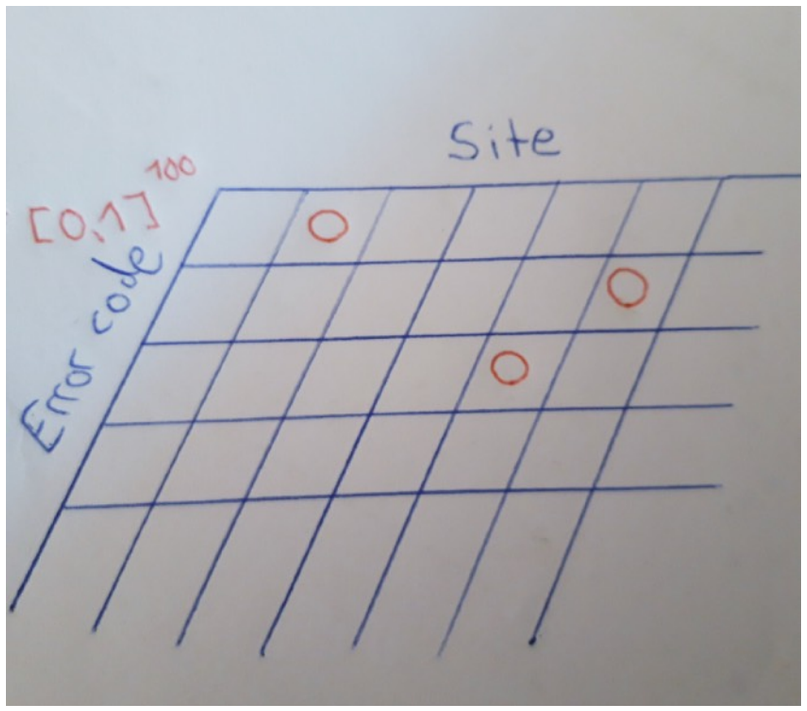
	task_name	site	error	error_msg	word2vec
0	/fabozzi_Run2016D-07Aug17-v1-SingleElectron-17...	T1_US_FNAL_Disk	85	Adding last 25 lines of CMSSW stdout: 775 fi...	[-0.0594909223050279,0.03154664652861862,-0.2...
1	/fabozzi_Run2017B-v1-DoubleMuon-09Oct2017_940p...	T1_RU_JINR	92	Adding last 25 lines of CMSSW stdout: %MSG-w X...	[0.18901344878493365,0.008977399177928554,-0.1...
2	/fabozzi_Run2017F-v1-DoubleEG-17Nov2017_940_17...	T2_UK_London_Brunel	99109	Error in StageOut: 99109 <@========== WMExcept...	[-0.0774647935680082,0.0897068816288741,-0.234...
3	/fabozzi_Run2017F-v1-DoubleMuon-09May2018_947_...	T2_US_MIT	8004	An exception of category 'BadAlloc' occurred w...	[-0.09825289702184319,0.21104744155693828,-0.3...
4	/fabozzi_Run2017H-v1-FSQJet2-17Nov2017_944_180...	T2_US_Nebraska	92	Adding last 25 lines of CMSSW stdout: [2018-04...	[0.16224175106636646,-0.019873988819217273,-0....

Word vectors from HDFS

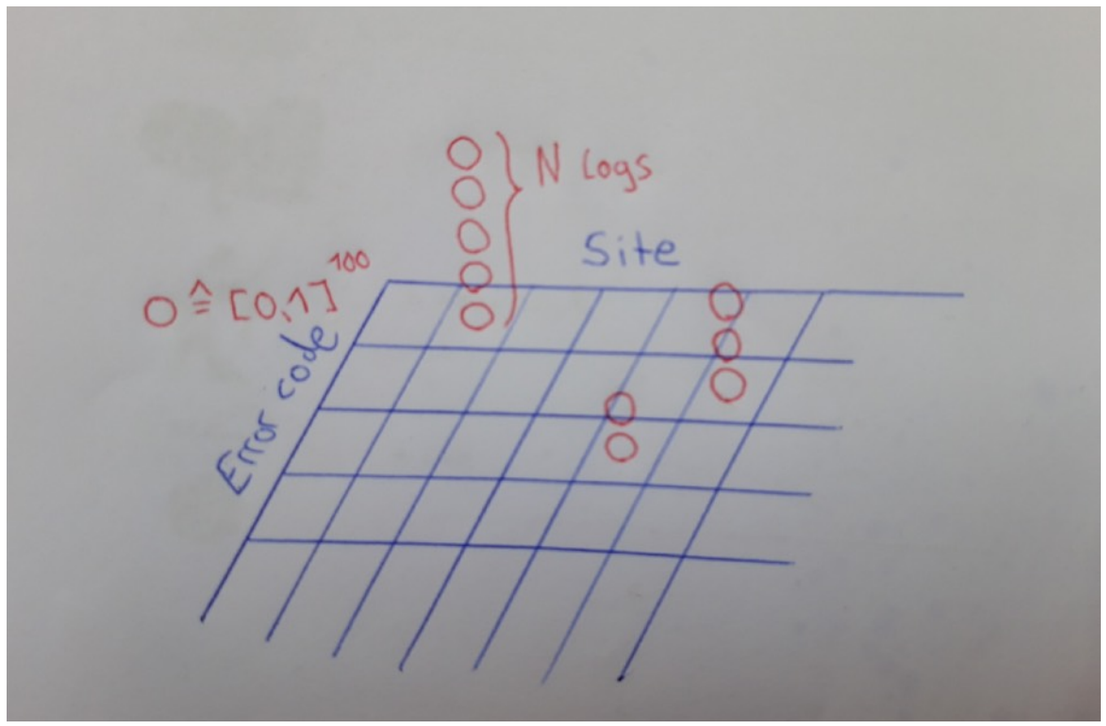
	task_name	errors	parameters	action	action_encoded	action_label
0	/amaltaro_Run2016D-v2-DoubleMuonLowMass-07Aug1...	{u'good_sites': {}, u'bad_sites': {u'-1': {u'T...	{u'action': u'acdc', u'sites': [u'T1_US_FNAL']...	acdc	0	acdc
1	/amaltaro_Run2016D-v2-DoubleMuonLowMass-07Aug1...	{u'good_sites': {}, u'bad_sites': {u'-1': {u'T...	{u'action': u'acdc', u'sites': [u'T1_US_FNAL']...	acdc	0	acdc
2	/amaltaro_Run2016D-v2-DoubleMuonLowMass-07Aug1...	{u'good_sites': {}, u'bad_sites': {u'-1': {u'T...	{u'action': u'acdc', u'sites': [u'T1_US_FNAL']...	acdc	0	acdc
3	/amaltaro_Run2018A-v1-DoubleMuon-17Sep2018_102...	{u'good_sites': {u'85': {u'T1_UK_RAL': 1}}, u'...	{u'action': u'acdc', u'cores': u'', u'xrootd':...	acdc	0	acdc
4	/amaltaro_Run2018A-v1-DoubleMuon-17Sep2018_102...	{u'good_sites': {u'-1': {u'T2_DE_RWTH': 1, u'T...	{u'action': u'acdc', u'cores': u'', u'sites': ...	acdc	0	acdc

Actionshistory (from Christian)

Input for the machine learning



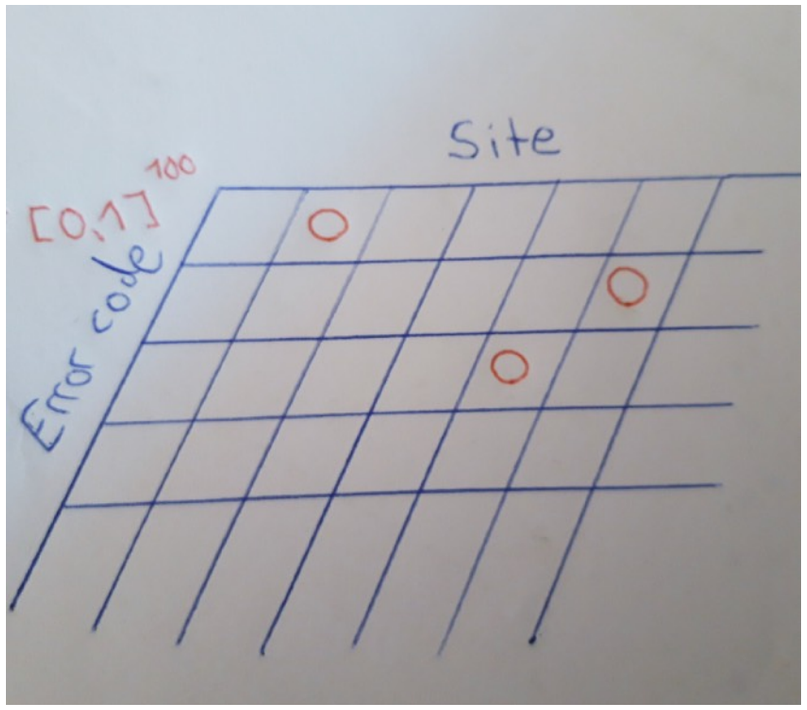
Input per workflow with one word vector per (task,site,error)



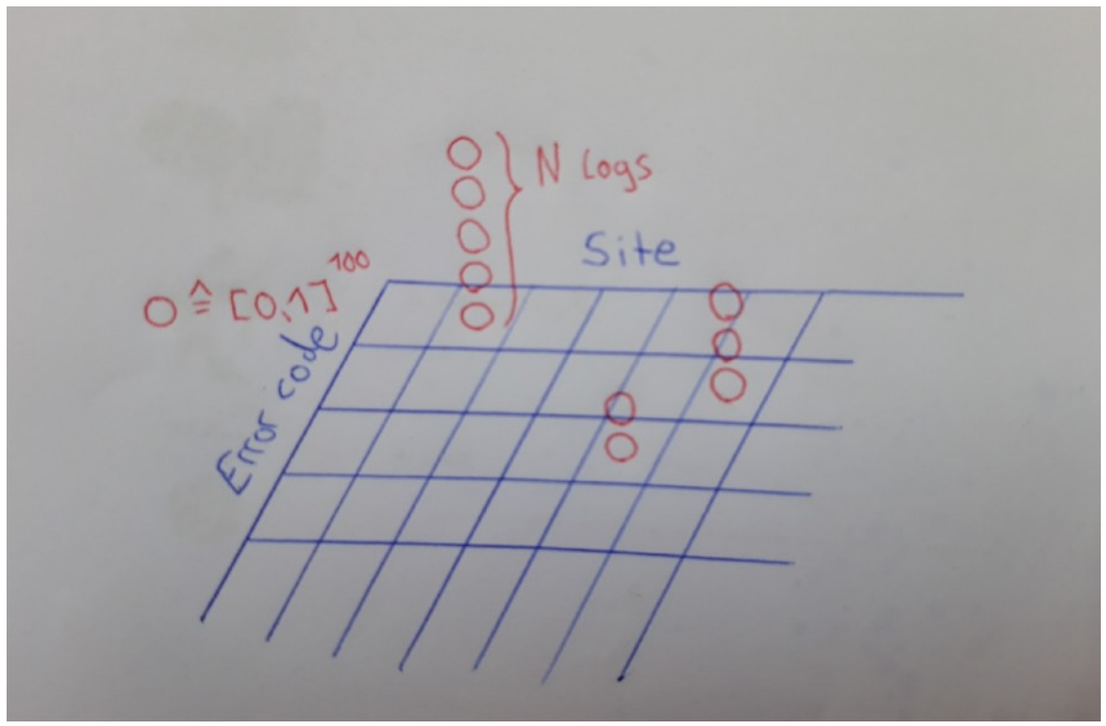
Input per workflow with multiple word vectors per (task,site,error)

- Complicated, high dimensional features
- Good hardware needed - hopefully avoid spark (but can also be done)
- Experiment with models (and ask experts) in the next weeks
- Build on Dominykas (and Hamed’s?) framework for k-fold evaluation

Input for the machine learning



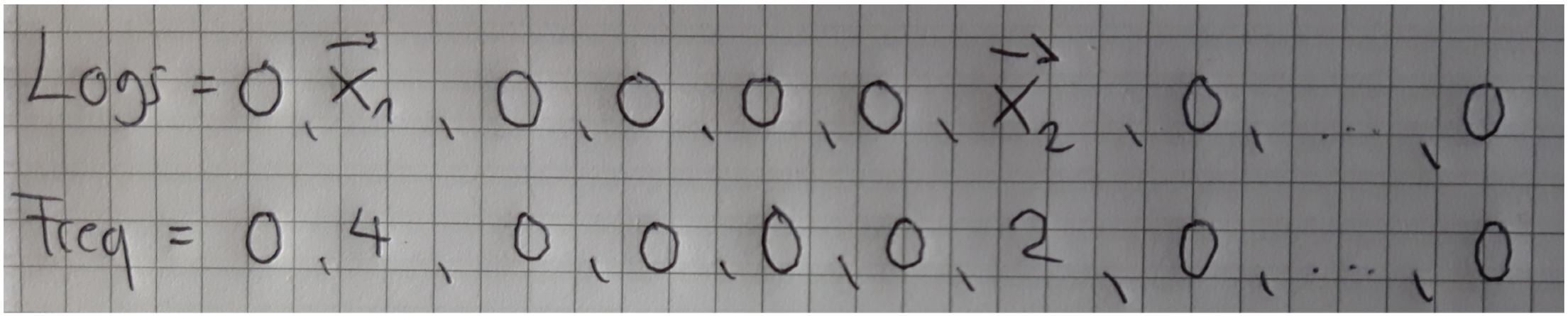
Input per workflow with one word vector per (task,site,error)



Input per workflow with multiple word vectors per (task,site,error)

- Flatten matrix + RNN?

- Merge with frequency of occurrences (standard of Christian)



Summary + Plans

Current status

- First prototype for ML input
- Concrete concept how to acquire the missing data

Next steps

- Set up a first model
- Validate the pipeline from EOS to HDFS

Plans

- Have a complete pipeline until end of June
- Optimize with experts at CERN during 2-3 months summer stay