Error log analysis with NLP

Lukas Layer

Overview

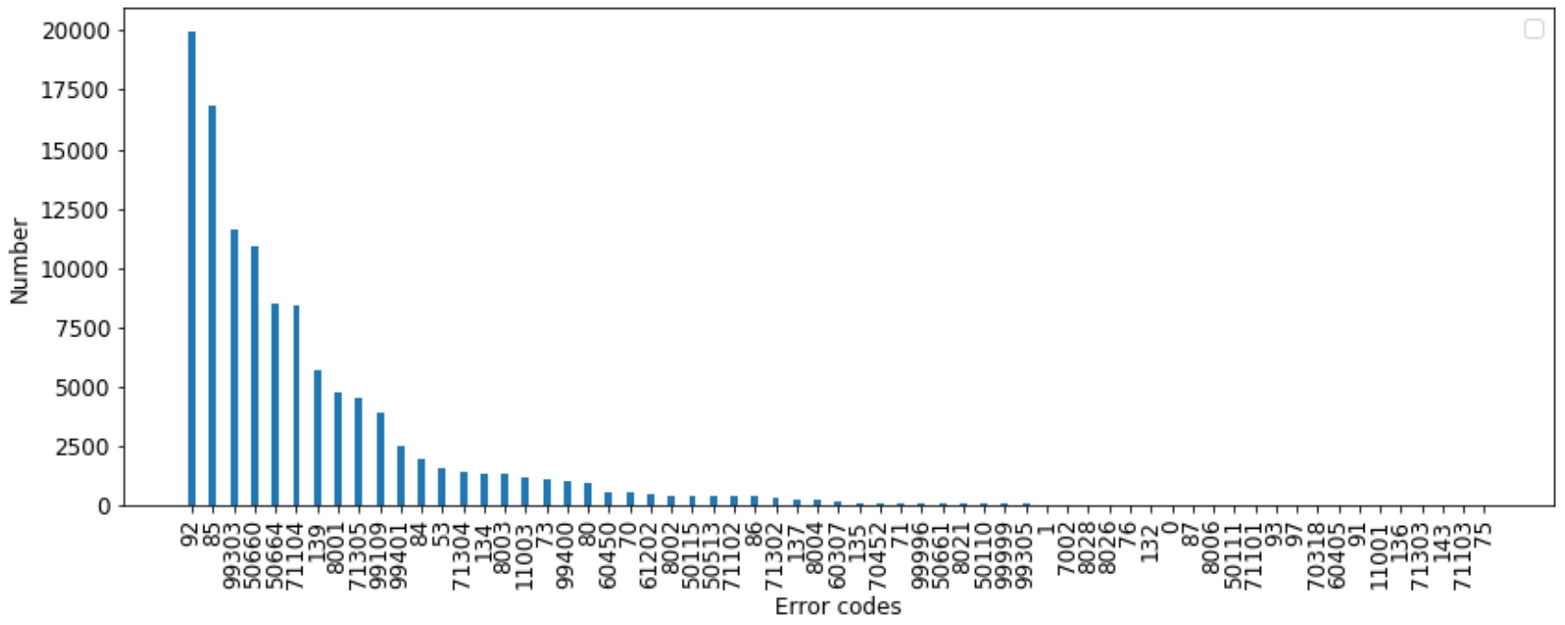
- Today: discuss NLP, preparation of the error message snippets from WMArchive and plans for the models

- Next time: discuss the results of the training of the models

- Main assumptions: only one error message per workflow, site and error + if a step in WMArchive entry has redundant error codes choose the first one that appears

- Repo https://github.com/llayer/AIErrorLogAnalysis

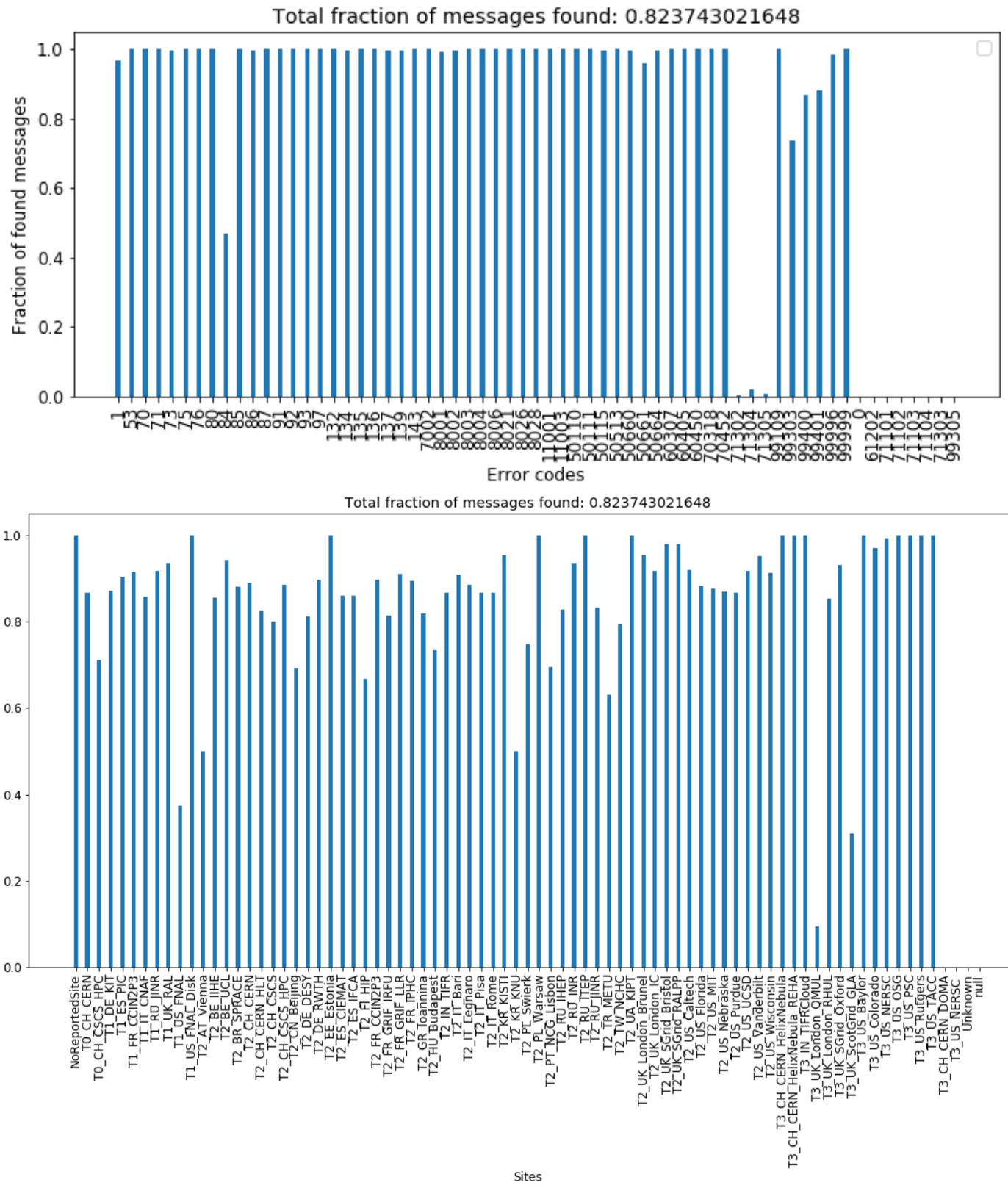
Analysis of actionshistory.json




```
Number of workflows 25090

All (task_name, site, error) keys
Number of all unique keys: 545863
Number of all unique sites: 151
Number of all unique errors: 64

All (task_name, site, error) keys with error != -1
Number of all unique keys: 114134
Number of all unique sites: 79
Number of all unique errors: 63

Most frequent error codes:
[u'92', u'85', u'99303', u'50660', u'50664', u'71104']
```

- 25.000 workflows with ~115.000 potential error messages
- ~50% of the sites only have -1 error codes
- Exit codes description: https://twiki.cern.ch/twiki/bin/view/CMSPublic/JobExitCodes

Found error messages in WMArchive



- Analysis done with spark
- Found ~ 82% error messages for the (task, site, error) keys in actionshistory.json
- Some error codes seem to not have messages reported

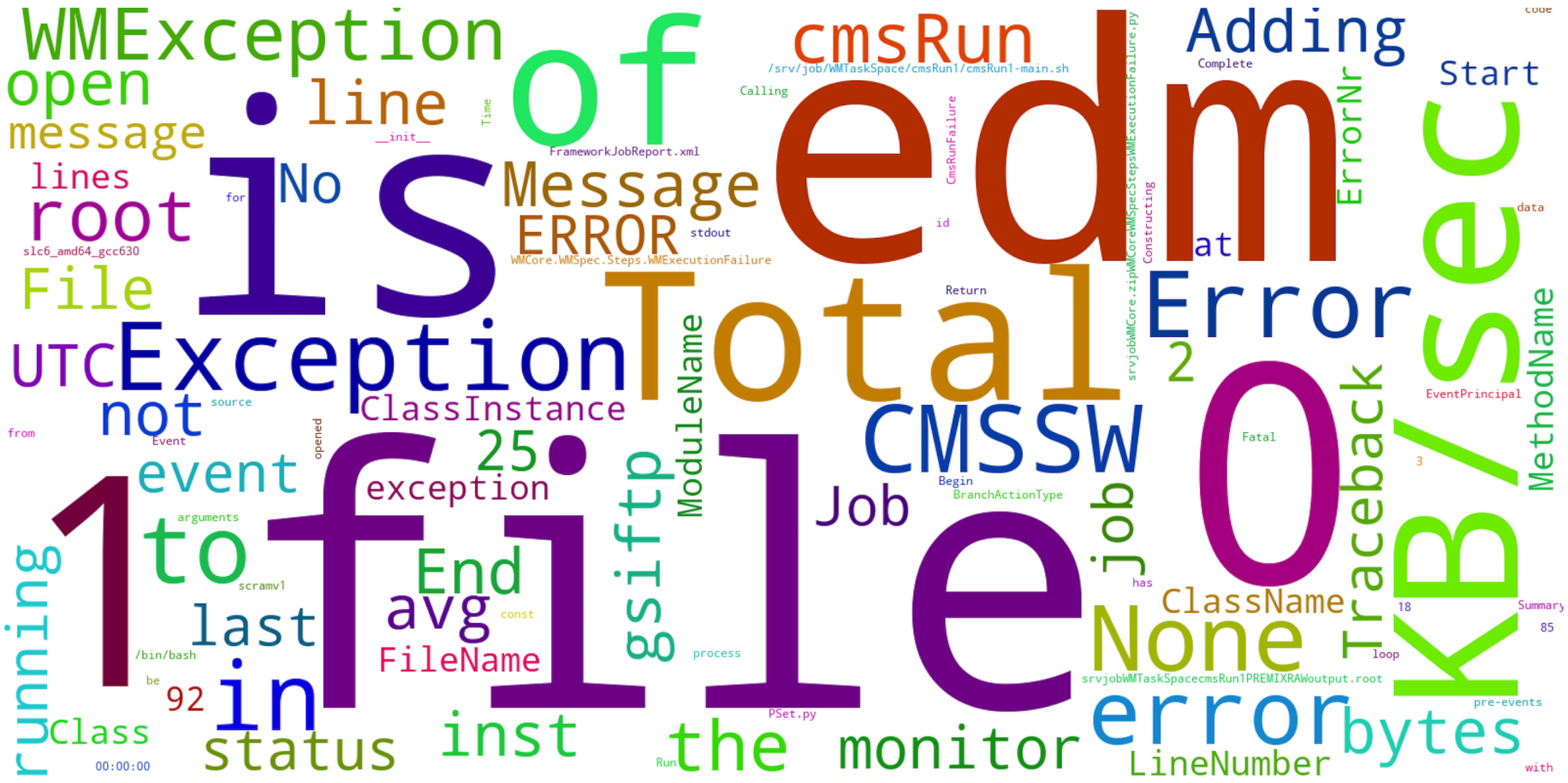
Preprocessing of the error logs

- Tokenize: NLTK TreeBankTokenizer (recommended in literature): statistical model trained on news text

- Filter: tokens that are only punctuation, single characters, frequency in whole vocabulary < 5

```
Original message

Adding last 25 lines of CMSSW stdout:   734 PFEGammaAlgo::mergeROsByAnyLink 1/115407         1/115529         1/11812
6   735 PFTrackTransformer   1/115277         1/115277         1/118020   736 PFTrackTransformer   1/115277         1
/115277         1/118020   737 RecoMuon             1/115516         1/117754         1/118143   738 XrdAdaptor
pre-events                          739 XrdAdaptorInternal   pre-events       pre-events       pre-events   740 XrdF
ileWarning       pre-events                            741 FailedPropagation    1/115534                              7
42 TimeReport           PostGlobalEndRun                      743 TrajectoryNotPosDef  1/115282
744 Fatal Exception      AfterSource                           745 Fatal Exception      1/118162         1/118154
1/118166   746 MemoryReport         PostGlobalEndRun                     747 fileAction           PostGlobalEndRun
748 fileAction           pre-events       pre-events         Severity    # Occurrences   Total Occurrences --------
-------------   ----------------- Warning             21574                21574 Error                   3
3 System                  9                    9  dropped waiting message count 5338 Complete process id is 224 status
is 85 <@========== WMException Start ==========@> Exception Class: CmsRunFailure Message: Error running cmsRun {'argu
ments': ['/bin/bash', '/srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh', '', 'slc6_amd64_gcc630', 'scramv1', 'CMSSW', 'C
MSSW_9_4_7', 'FrameworkJobReport.xml', 'cmsRun', 'PSet.py', '', '', '']} Return code: 85        ModuleName : WMCore.W
MSpec.Steps.WMExecutionFailure  MethodName : __init__   ClassInstance : None    FileName : /srv/job/WMCore.zip/WMCore
/WMSpec/Steps/WMExecutionFailure.py     ClassName : None        LineNumber : 18         ErrorNr : 85  Traceback:   <@
---------- WMException End ----------@>

Filtered

['Adding', 'last', '25', 'lines', 'of', 'CMSSW', 'stdout', '734', 'PFEGammaAlgo', 'mergeROsByAnyLink', '735', 'PFTrac
kTransformer', '736', 'PFTrackTransformer', '737', 'RecoMuon', '738', 'XrdAdaptor', 'pre-events', '739', 'XrdAdaptorI
nternal', 'pre-events', 'pre-events', 'pre-events', '740', 'XrdFileWarning', 'pre-events', '741', 'FailedPropagation'
, '742', 'TimeReport', 'PostGlobalEndRun', '743', 'TrajectoryNotPosDef', '744', 'Fatal', 'Exception', 'AfterSource',
'745', 'Fatal', 'Exception', '746', 'MemoryReport', 'PostGlobalEndRun', '747', 'fileAction', 'PostGlobalEndRun', '748
', 'fileAction', 'pre-events', 'pre-events', 'Severity', 'Occurrences', 'Total', 'Occurrences', 'Warning', 'Error', '
3', '3', 'System', '9', '9', 'dropped', 'waiting', 'message', 'count', 'Complete', 'process', 'id', 'is', '224', 'sta
tus', 'is', '85', 'WMException', 'Start', 'Exception', 'Class', 'CmsRunFailure', 'Message', 'Error', 'running', 'cmsR
un', 'arguments', '/bin/bash', '/srv/job/WMTaskSpace/cmsRun1/cmsRun1-main.sh', 'slc6_amd64_gcc630', 'scramv1', 'CMSSW
', 'CMSSW_9_4_7', 'FrameworkJobReport.xml', 'cmsRun', 'PSet.py', 'Return', 'code', '85', 'ModuleName', 'WMCore.WMSpec
.Steps.WMExecutionFailure', 'MethodName', '__init__', 'ClassInstance', 'None', 'FileName', 'srvjobWMCore.zipWMCoreWMS
pecStepsWMExecutionFailure.py', 'ClassName', 'None', 'LineNumber', '18', 'ErrorNr', '85', 'Traceback', 'WMException',
'End']
```

Wordcloud for the 100 most frequent words



- ~ 57.000 Tokens after filter
- Harder filtering possible - remove stopwords, lower, stemming …

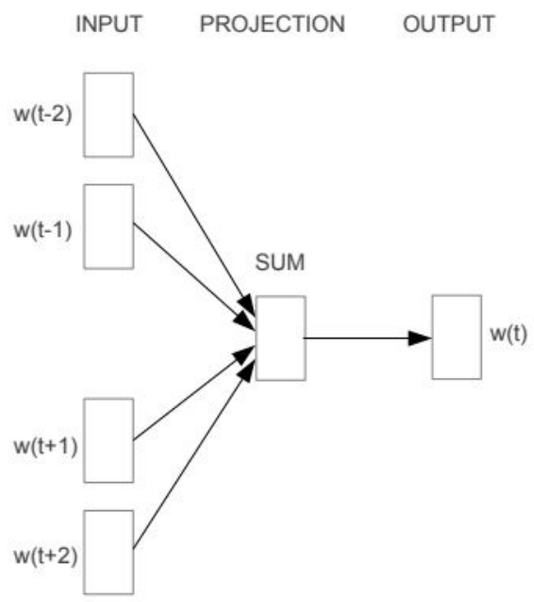
Word2Vec

- Vector space models: embed words in a continuous vector space where semantically similar words are mapped to nearby points -> capture relations and reduce dimensionality

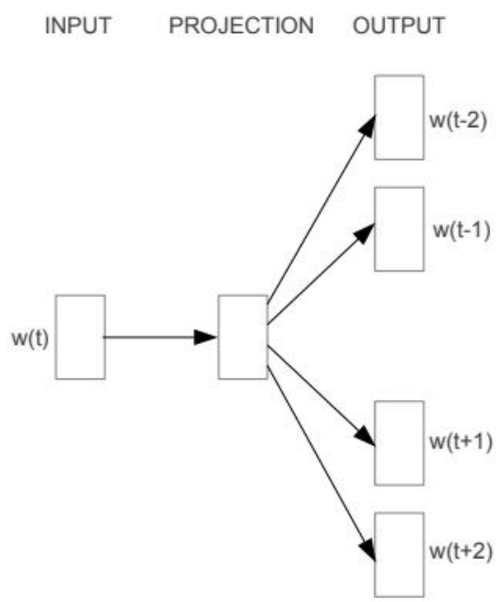
- Word2vec: particularly computationally-efficient predictive model for learning word embeddings from raw text https://www.tensorflow.org/tutorials/representation/word2vec

- Skip-Gram Model: learning word embeddings by training a model to **predict context given a word**

- Continuous Bag of Words (CBOW) model: learning word embeddings by training a model to **predict a word given its context**



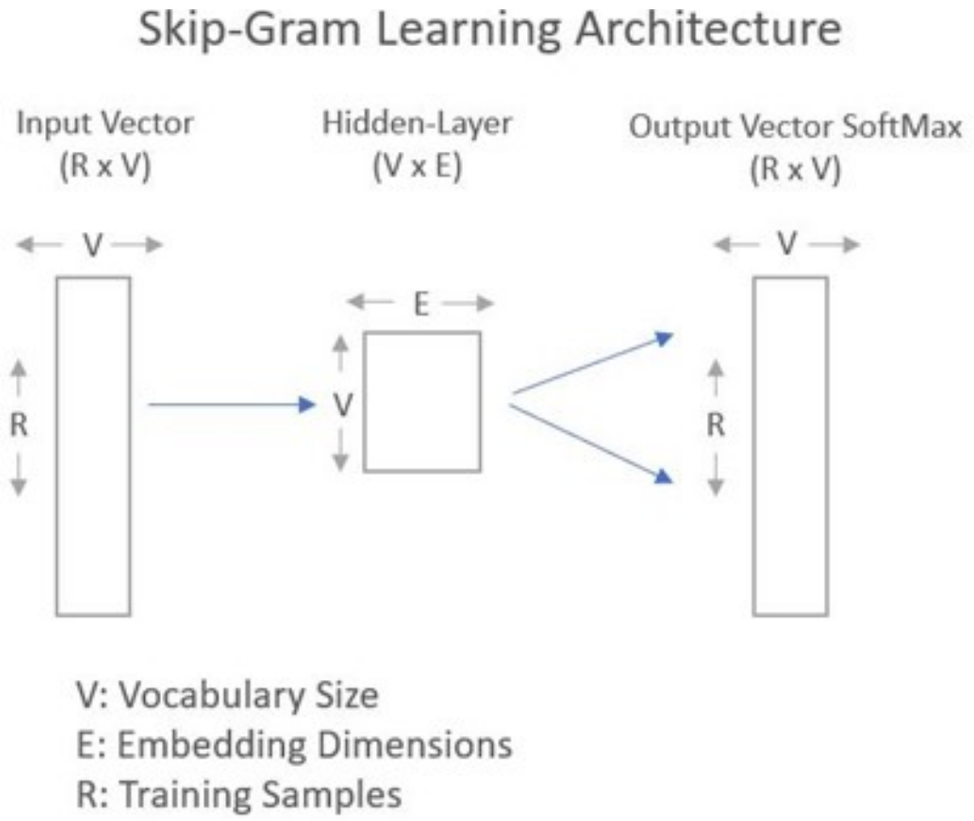
CBOW



Skip-gram

3 Layer NN with input, hidden and output layer

Skip-Gram algorithm



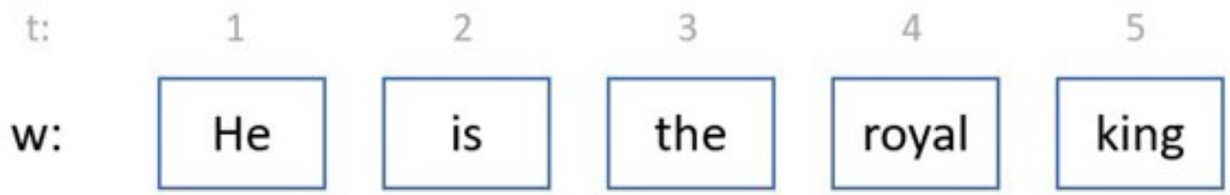
- Input layer: vocabulary (R x V) vector in which V=Vocabulary Size and R is the number of training samples

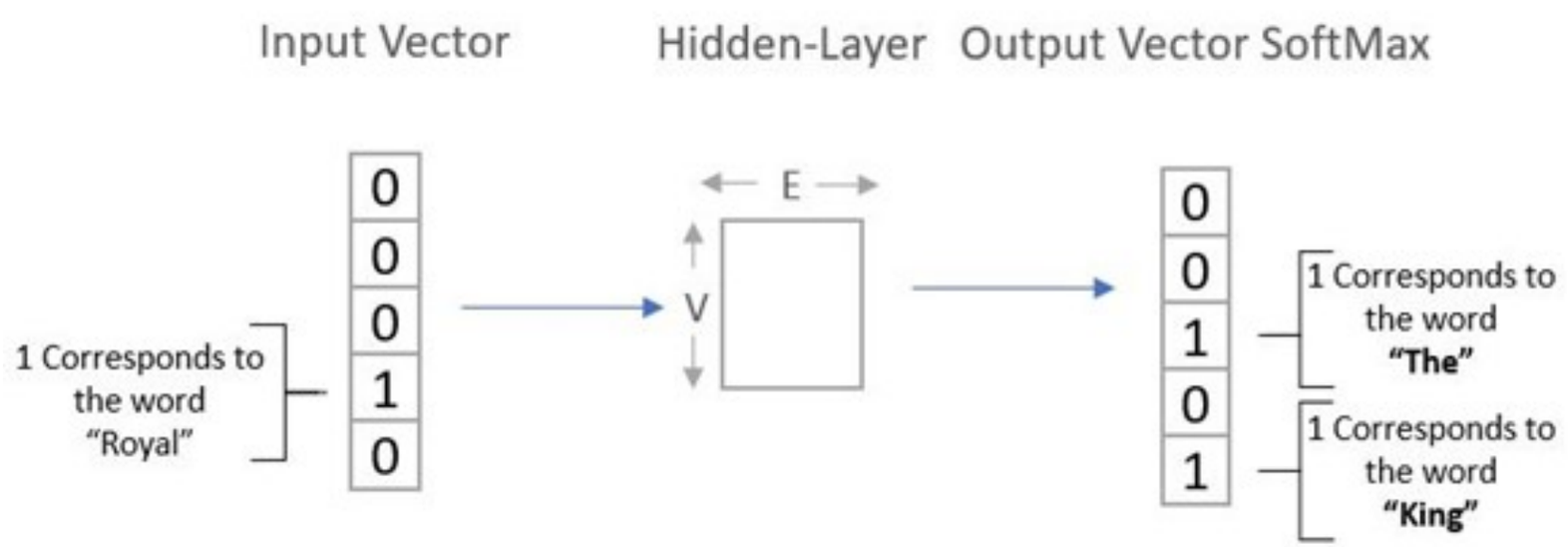
- Each word in the vocabulary is represented by a one-hot encoded vector

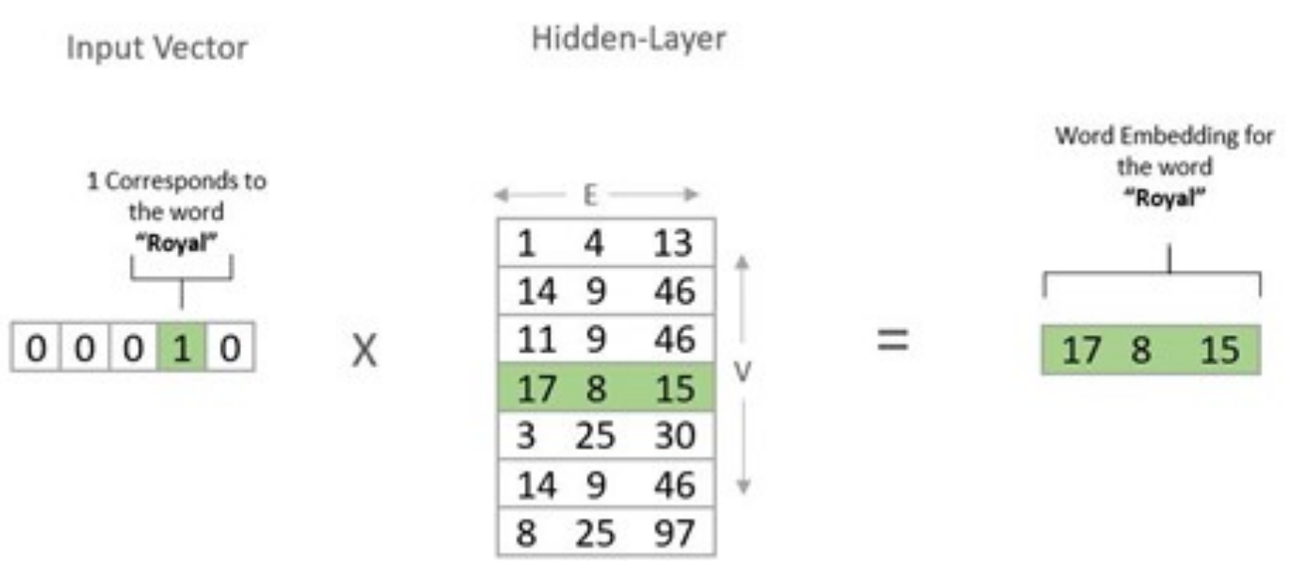
- Hidden layer: (V x E) in which E = Embedding Dimensions

- The output layer is a vector (R x V). Probability for each word in the vocabulary for the given word input -> softmax

Skip-Gram example

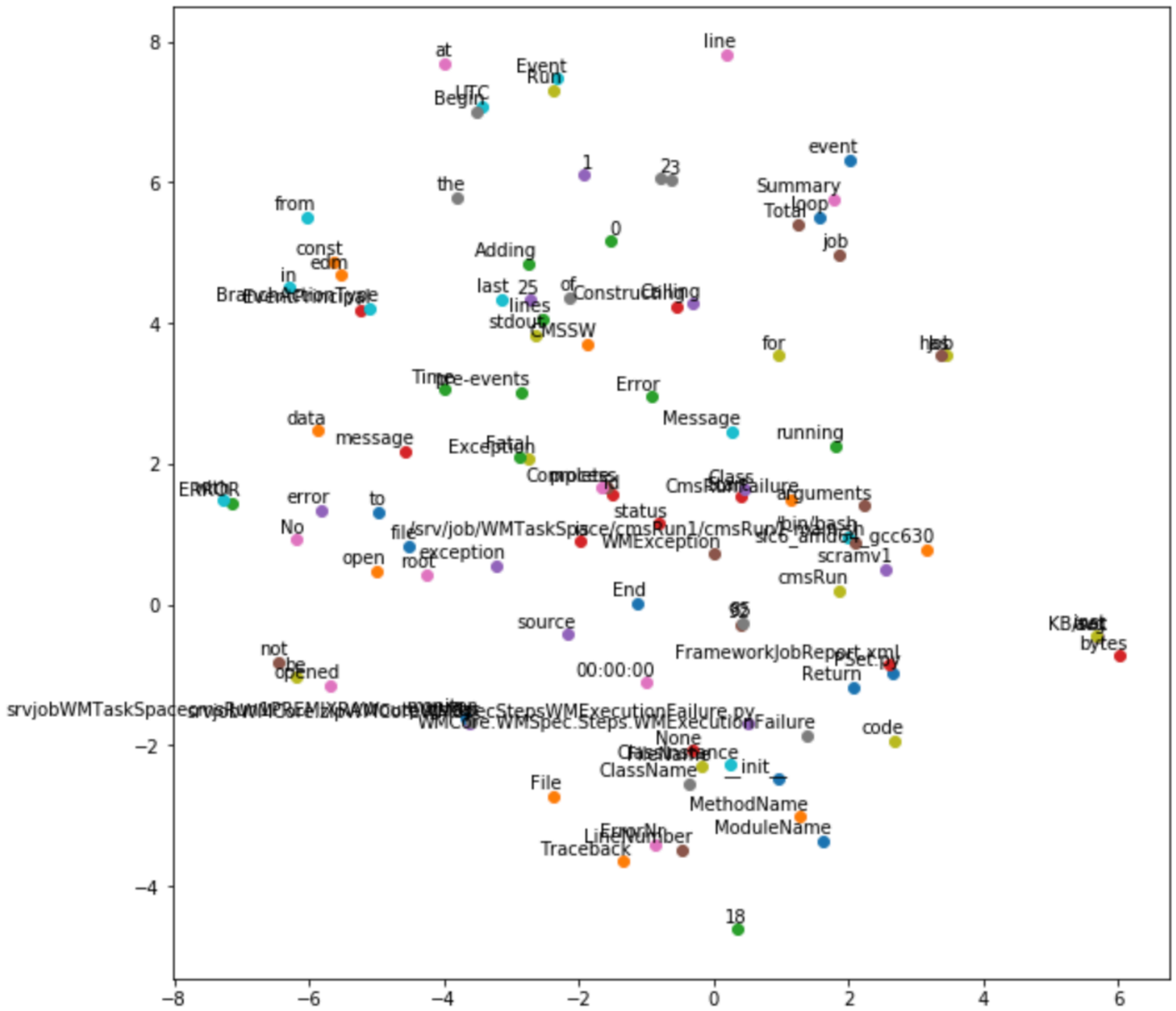




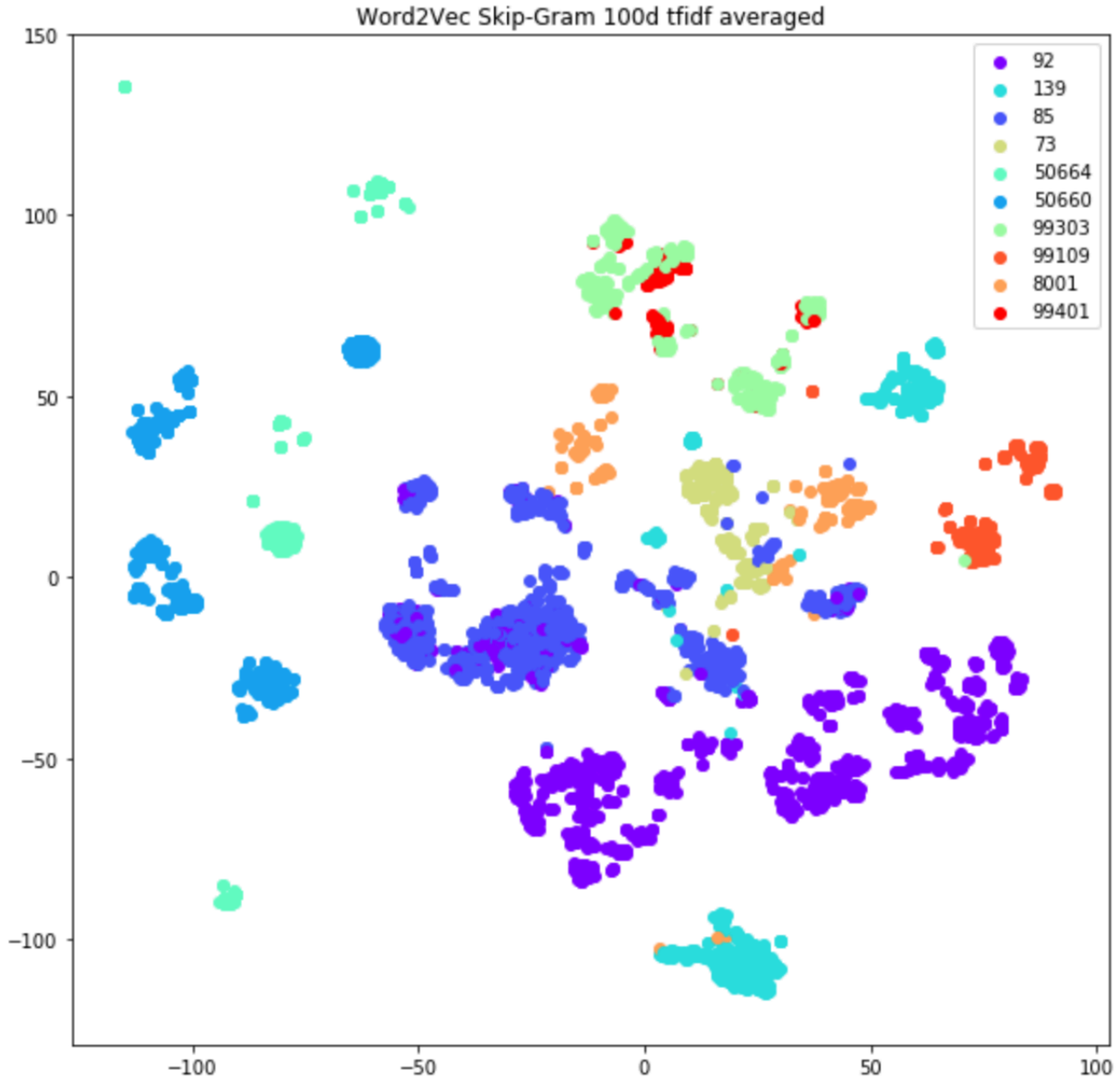


Word2vec for the error messages vocabulary

- Train with skip-Gram in 100d
- Use t-SNE algorithm to visualize embeddings
- t-SNE: nonlinear dimensionality reduction -> similar objects are modeled by nearby points and dissimilar objects are modeled by distant points
- Fraction in Google news: < 5%

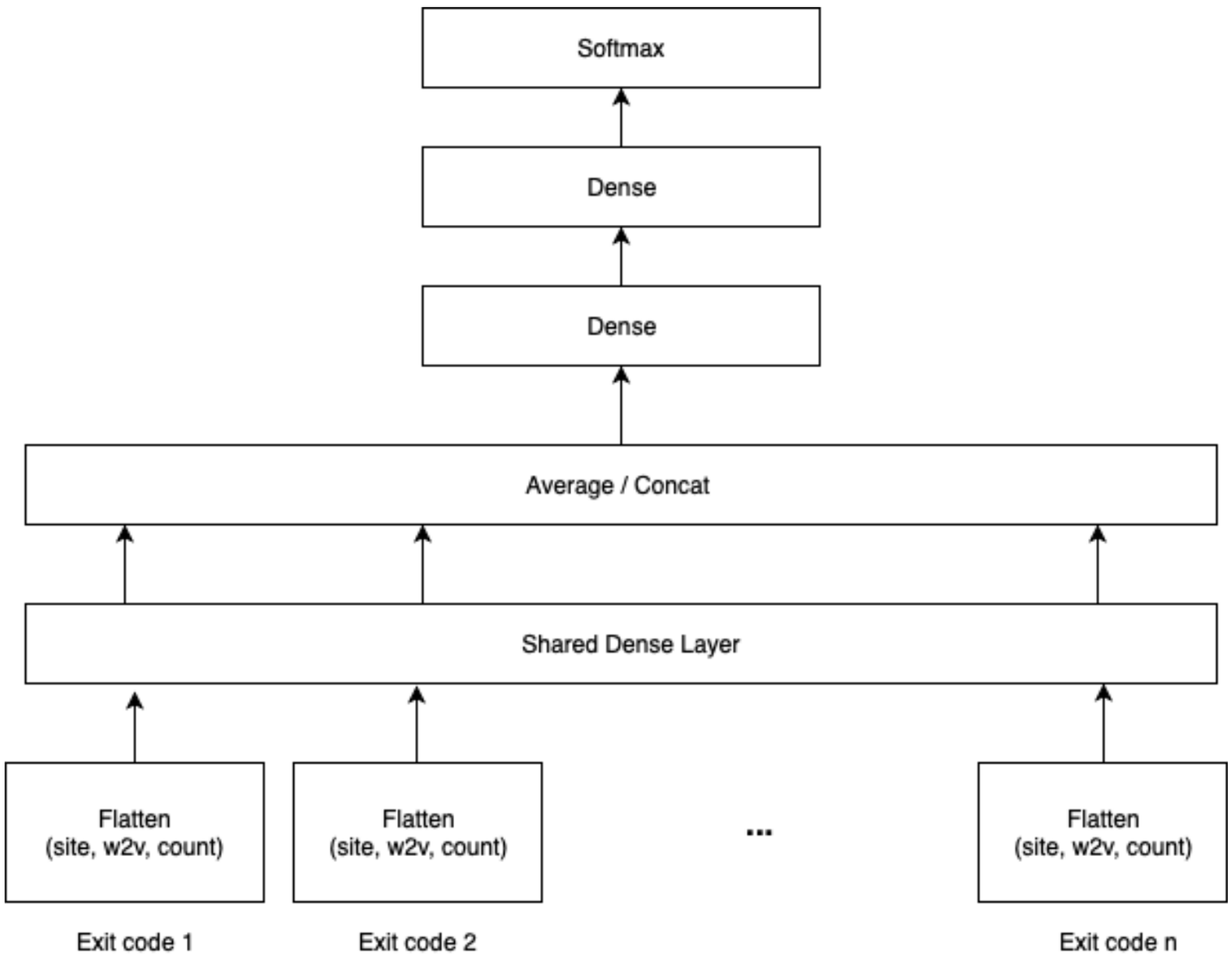


Visualization of the error messages with t-SNE



- Representation of the error message -> average text vectors
- More advanced: weight with TF-IDF: **term frequency–inverse document frequency:** reflects how important a word is to a document in a corpus
- Error codes form clusters

Model sketch to include pretrained word vectors



- Input numpy array: (workflows, exit_code, site, w2v + count) -> (25.000 x 50 x 100 x 100 x 1)
- Fully connected NN has too many parameters -> use shared layers and variations
- Setup the matrix speeding up Christian's code using pandas and filling a 3D numpy array per task-> O(10s) to create input
- In case of memory issues: create matrix in chunks and write to hdf files -> use Keras fit generator
- Prototype implemented and tested

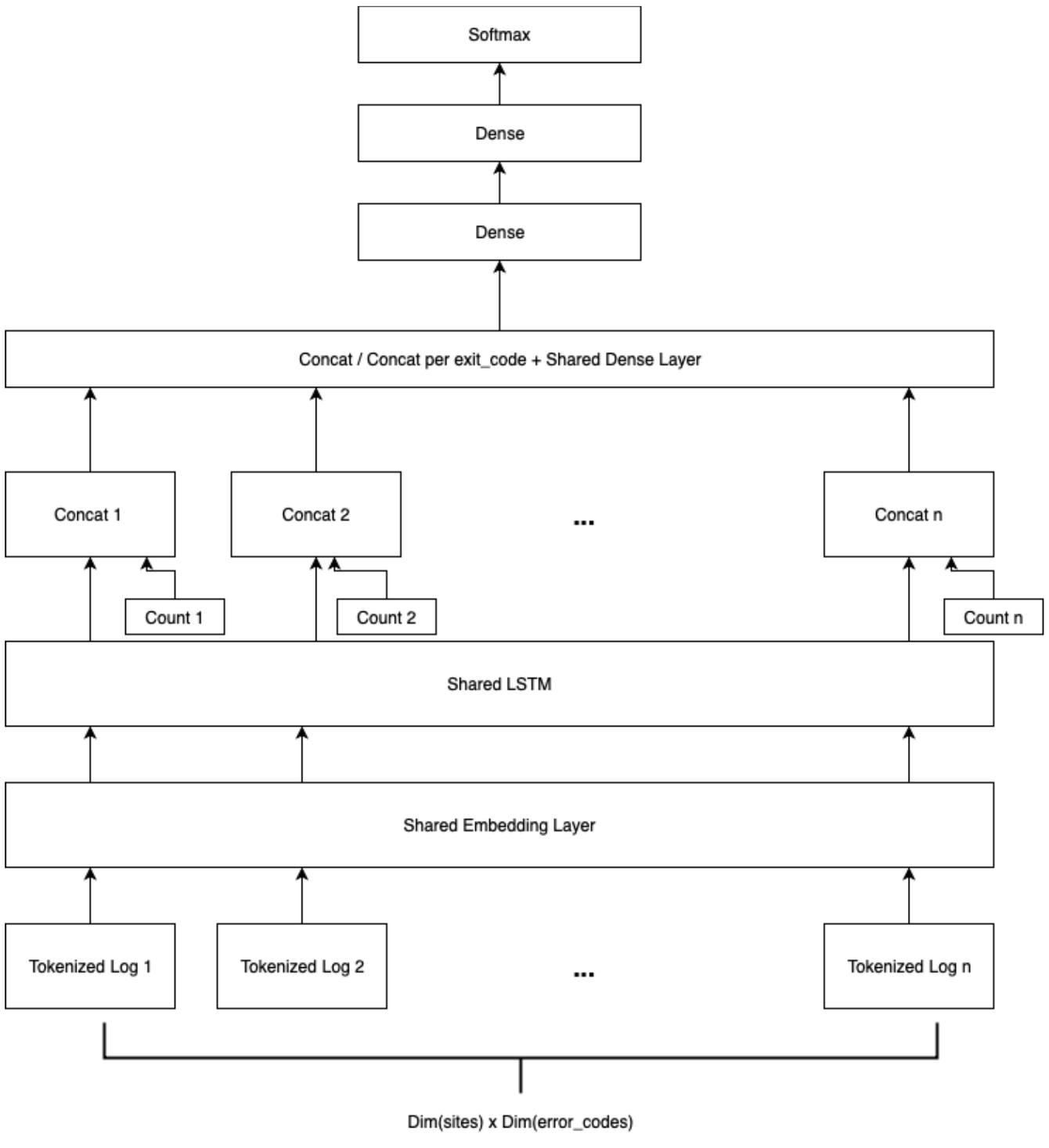
More advanced: include text directly in model

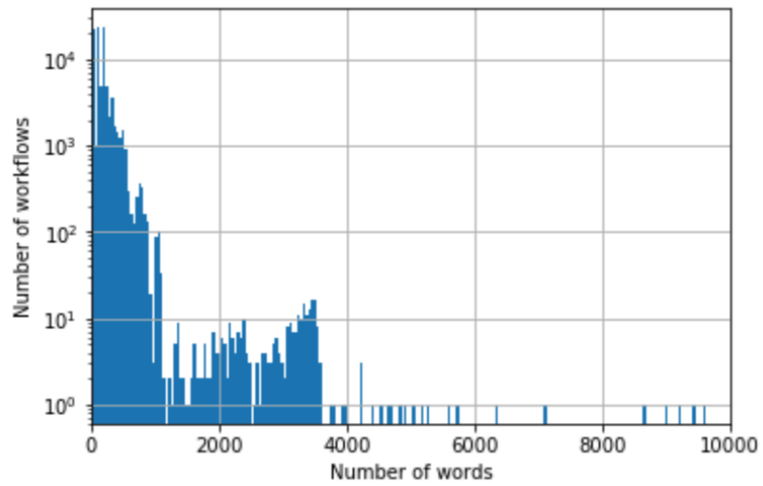
```
# Build the model
embedding_vector_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam', metrics=['accuracy'])
print(model.summary())
```

Example: sentiment analysis - predict if review for a movie is positive/negative

- Problem with averaging word vectors -> no exploitation of the sequence of words
- Solution: include Embedding Layer directly: input_dim = number of words, output_dim = embedding dimension, input_length = maximum length of sentence
- Embedding Layer reduces dimensionality - avoids one hot encoding of words + can be initialized with pretrained models - transfer w2v model
- LSTM: used for sequences -> feed the sentence
- Review is positive or negative -> binary_crossentropy

Model sketch to include the tokenized logs directly





- Include Embedding + LSTM per (error, site) matrix cell

- Challenge: dimension is now: (25.000 x 50 x 100 x ~5.000)

- Words in error messages are encoded by integers and need to be masked with zeros -> very sparse, hard to write to disk

- Limit with Keras: fit one example in the GPU memory

- Summing over sites/errors also possible

Pipeline for the ML

Training + Evaluation

- Build on Dominykas code -> includes k-fold validation, bayesian optimization and oversampling

- Need to be adapted to out of memory training -> load the data in chunks and use Keras fit_generator (prototyped)

Possible solution to train a model as shown in the last slide

- Store input per workflow as a sparse matrix with dictionary { (site_index, error_index) : [tokenized text], … }

- Fits easily in memory -> Loop over the workflows, create inflated matrix with masking zeros in chunks and train the model in batches

- Another possibility: Keras accepts sparse input -> try to bring the sparse input in a form that Keras can handle

Follow up: multiple errors per WMArchive entry


```
{ …,
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', }], …
}
```


- Multiple steps, error reporting not optimal, initial error can cause more error further the chain - schema: https://github.com/dmwm/WMCore/blob/master/src/python/WMCore/Services/WMArchive/DataMap.py#L378

- What is the actionshistory.json?

- What is the best way to deal with redundant exit codes? Concat, first one… ?

Open questions / ideas

- Best way of combining with Hamed’s code
- Add good/bad sites information to training?
- Sum over Tiers, Sites, etc.
- Add info from other fields in WMArchive
- Make structured templates from log messages

Summary + Plans

Current status

- First pipeline to train and evaluate a model implemented
- Several ideas for serious models

Plans for the next weeks

- Verify the assumption that multiple error messages per (task, site, error) are redundant
- Better understand + improve the filtering of the vocabulary (define working points)
- Finish implementing the models and handle the training and evaluation for out-of-memory data (adapting Dominykas code)
- Run the proposed models under equal conditions to have a fair comparison

Long-term plans

- Merge code with Hamed
- Switch to TF
- Include the full error logs to have more control