Using Gensim to do Latent Semantic Indexing of Text

Misty Nodine

September 11, 2013

Text Data is Messy

Salary Prediction: Data Set Features
http://www.kaggle.com/c/job-salary-prediction/data

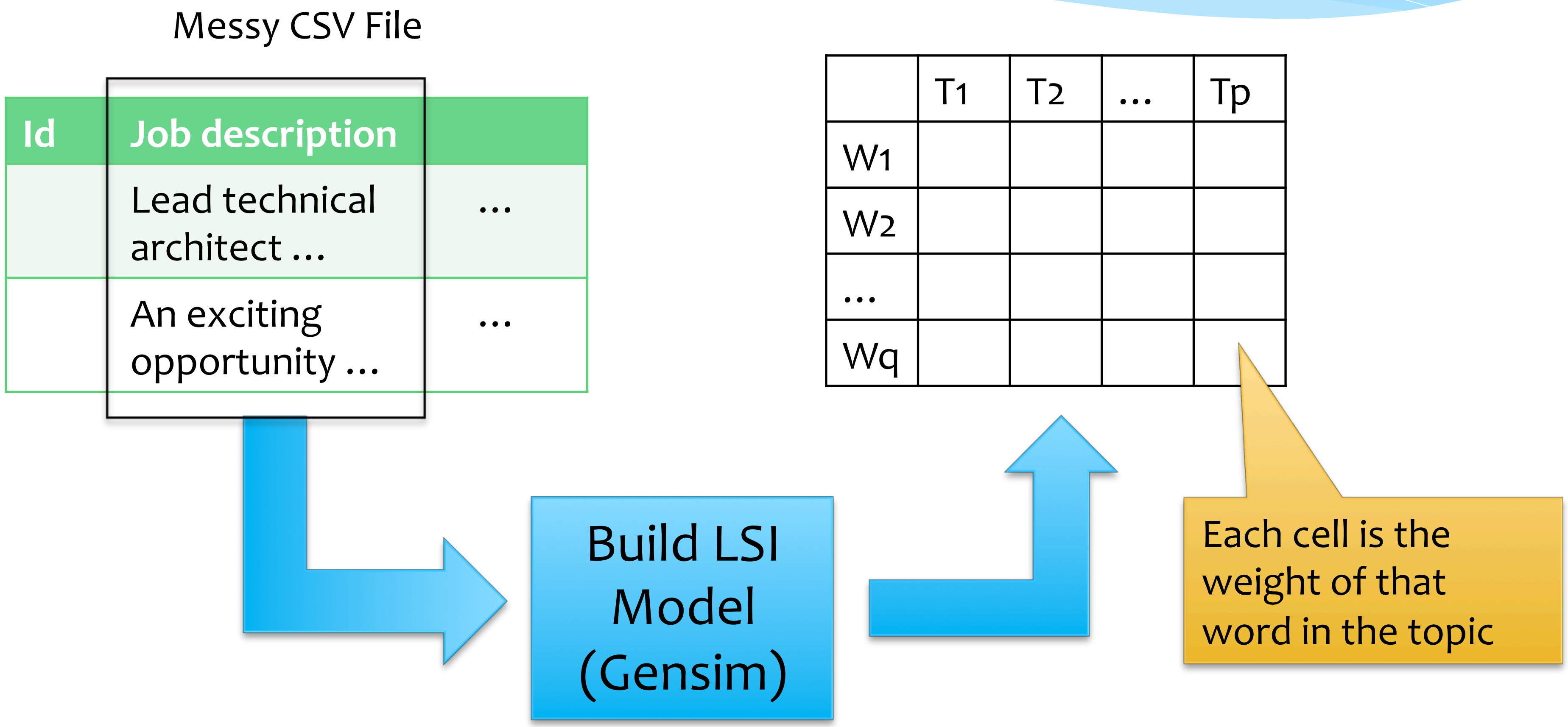
Id	Title	FullDescription	Location	Company
13656201	Lead Technical Architect, C Banking	Lead Technical Architect required for a Tier **** Investment Bank with excellent C skills. The main function of the role is to be the architectural lead, in particular designing solution architecture that will support the strategic vision. Draft the roadmap ...	London	Scope AT Limited
27754964	Software Developers (All Levels)	An exciting opportunity for skilled and motivated C++ and Java Software Developers has arisen within a Software and Consultancy services company based in Cambridge. The candidates will join a team of developers to help develop ...	Cambridge	Indigo 21 Ltd
30292881	WEB Developer requires both front and back end developers	WEB Developer requires both front and back end developers Salary up to **** Farnham Surrey Our client is located in Farnham Surrey and are looking for Web developers both	Surrey	Gregory Martin International

Making Messy Text Data Tidy

- Using your training data, extract a fixed set of “topics” from the text.
- For each entry in the training set, compute the similarity between the text and each of the topics.
- This array of floats becomes the training features used for that text field when training your model on that entry.
- When classifying / understanding a new entry, compute its similarity to the set of topics to generate its features.

- We will be using Python 2.7 and the Gensim package.
- Gensim has several options for computing topics. We will be using LSI over TFIDF.

Topic Computation



Messy CSV File
Id
Job description
Lead technical architect …
…
An exciting opportunity …
…
Build LSI Model (Gensim)
T1
T2
…
Tp
W1
W2
…
Wq
Each cell is the weight of that word in the topic

The Dictionary of Words

```
from gensim import corpora

def createDictionary(input_filename):
    dictionary = corpora.Dictionary(tokenize(line) \
                      for line in open(input_filename))
    # Remove words that appear only once
    once_ids = [tokenid for tokenid, docfreq in \
                    dictionary.dfs.iteritems() if docfreq == 1]
    dictionary.filter_tokens(once_ids)
    dictionary.compactify()          # remove gaps in word ids
    return dictionary
```

Bag of Words

```
doc = "We currently operate over 5000 websites. We are
looking  for a system administrator to manage the servers.”

# Tokens are: ['currently', 'operate', 'over', 'websites',
'are', 'looking', 'for', 'system', 'administrator', 'to',
'manage', 'servers']

bow = dictionary.doc2bow(tokenize(test_doc, stoplist))
# Bag of words is: [(2934, 1), (3402, 1), (5712, 1), (6896,
1), (12279, 1), (14326, 1), (14928, 1), (15743, 1), (21330,
1), (21475, 1), (25093, 1), (25496, 1)]
```

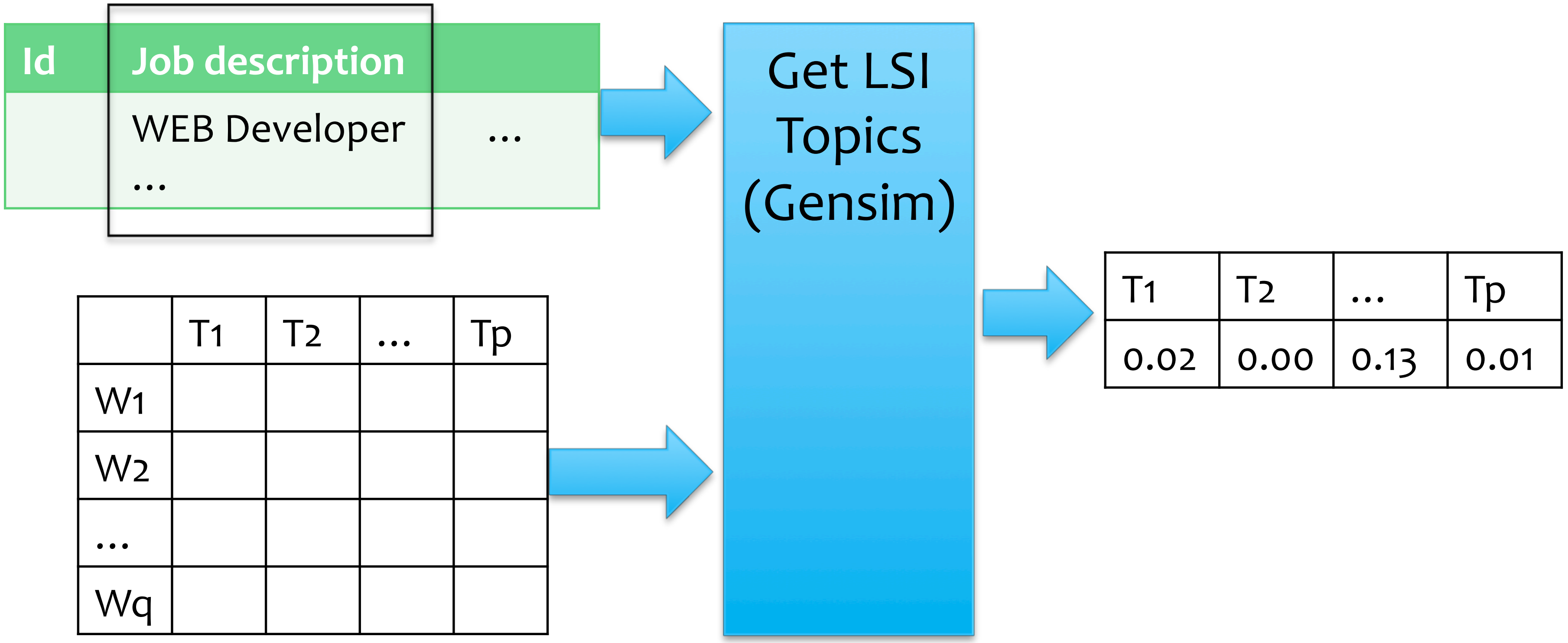
Computing the Models

```
from gensim import models

def getBows(filename, dictionary):
    corpus = [tokenize(line) for line in open(filename)]
    return [dictionary.doc2bow(text) for text in corpus]

def getTfidfModel(bows):
    return models.TfidfModel(bows)

def getLSIModel(bows, corpus_tfidf, dictionary, topic_count):
    return models.LsiModel(corpus_tfidf[bows], \
           id2word=dictionary, num_topics=topic_count)

corpus_bows = getBows(“./data/Train_IT.csv”, dictionary)
tfidf_model = getTfidfModel(corpus_bows)
corpus_lsi = getLSIModel(corpus_bows, \
                            tfidf_model, dictionary, 200)
```

What an LSI Topic Looks Like

```
Topic 0 is: [(0.22658075980725881,  'net'),
             (0.14218184673368597,  'developer'),
             (0.1223306965157543,   'asp'),
             (0.11933616192619613,  'software'),
             (0.1127069193752397,   'web'),
             (0.10985619748257568,  'sql'),
             (0.098785093501989552, 'support'),
             (0.095997020915663339, 'server'),
             (0.093096872247903806, 'development'),
             (0.088535058440318379, 'project')]
```

Find the Topics in a Document



Id	Job description	
	WEB Developer …	…

	T1	T2	…	Tp
W1				
W2				
…				
Wq				

T1	T2	…	Tp
0.02	0.00	0.13	0.01

Finding the Topics in a Document

```
def text2lsi(text, dictionary, tfidf_model, lsi_model):
    tokens = dictionary.doc2bow(tokenize(text))
    return lsi_model[tfidf_model[tokens]]
```

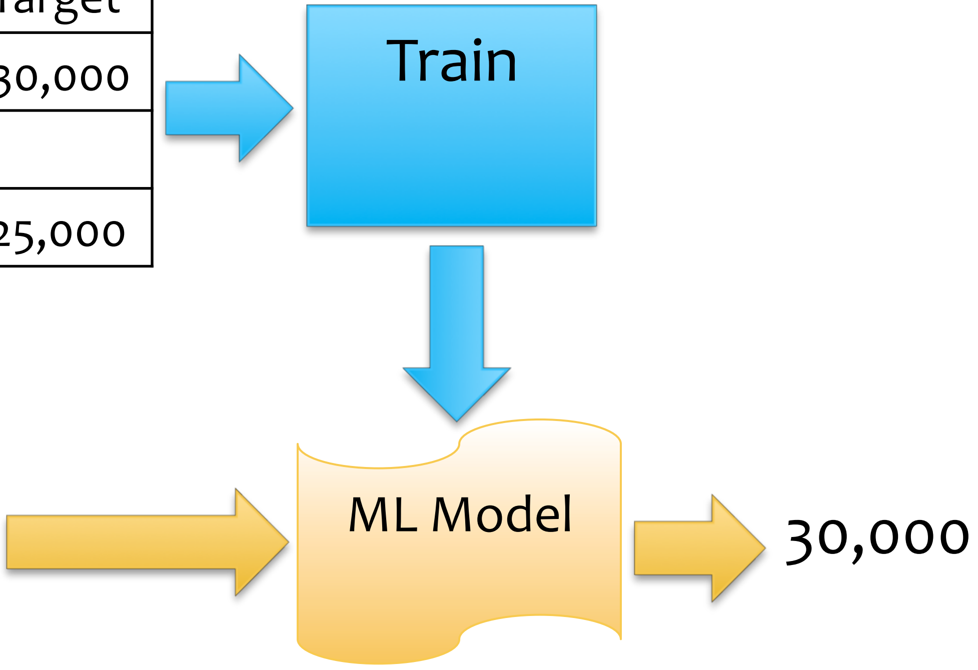
"Company XYZ is actively seeking .net software developers to work on our backend servers. Proficiency in SQL is a requirement."

```
First few LSI topics are: [(0, 0.11720663638483562),
                           (1, -0.1253705309432416),
                           (2, 0.0060785884779191689),
                           (3, -0.057803584368499326),
                           (4, -0.00084847871462808886),
                          …]
```

Using Topics for ML

Id	T1	T2	…	Tp	…	Target
a	0.02	0.00	0.13	0.01		30,000
…						
b	0.15	0.06	-0.1	0.02		25,000

Id	T1	T2	…	Tp	…
x	0.01	0.00	0.07	0.02	



Additional tutorial information and code are available in my iPython notebook.

https://github.com/mistynodine/PythonModules/LsiTutorial/tree/master/Mistys_LSI_tutorial.ipynb

Misty Nodine
nodine@alum.mit.edu