Dynamic Lexicon Generation for Natural Scene Images

Avinash Kumar
Gaurav Mahajan
Harshit Sharma
Pratyushotpal Madhukar

Presentation Outline

1. Project Overview
2. Flow Diagram
3. Datasets and Frameworks
4. Our Implementation
5. LDA Algorithm
6. CNN and Inception Network
7. Transfer Learning
8. Applications / Extension Ideas

Project Overview

1. Current methods approach scene text recognition problem from a word spotting perspective and assumes customized lexicons as given.

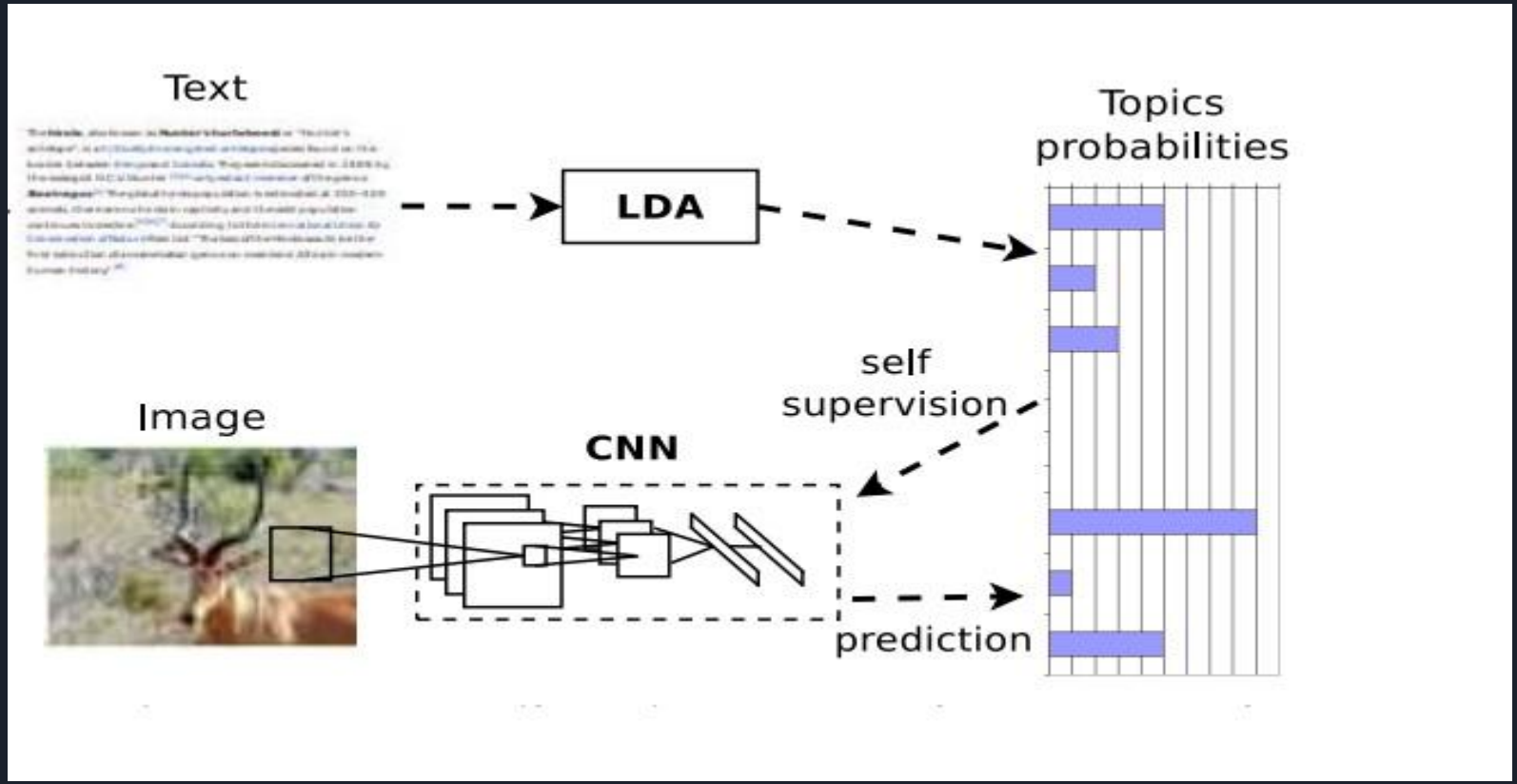
2. In real life scenarios lexicons need to be dynamically generated.

3. The project aims at generating contextualized lexicons dynamically just using visual information.

Why Dynamic Lexicon?

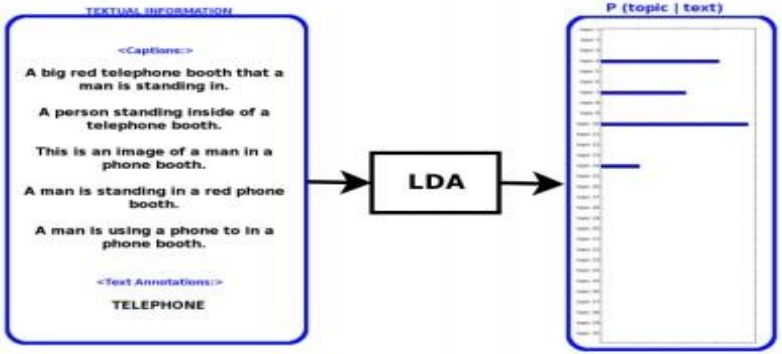
- Recognition performance drops on open vocabulary search space.
- Even in fixed search space re-ranking of words is not possible.
- The correlation between visual and textual information in a dataset consisting of images and textual content is exploited

Flow Diagram

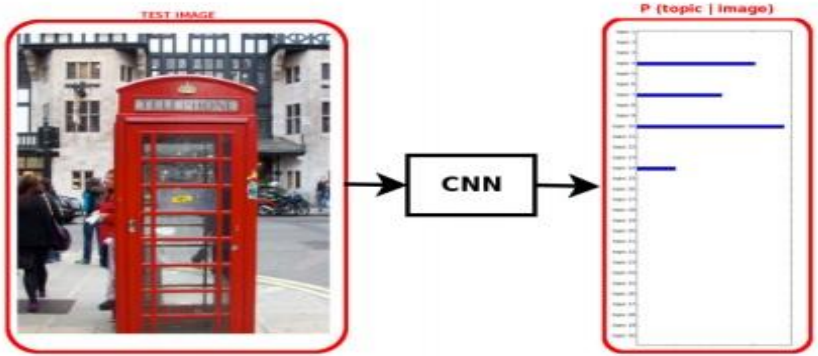


Flow Diagram Cont.

(a)



(b)



Datasets and Frameworks

Datasets:

1. MS-COCO
2. COCO-Text

For training purposes we use the training and validation sets of MS-COCO but removing the images that are part of the validation set in COCO-Text.
For validation purposes we use the validation set of COCO-Text.

Frameworks:
For **LDA**: Gensim
For **CNN:** TensorFlow

Step 1 : Generating Dataset Corpus

Input : file containing images captions and annotations (actual words in image)

Output : JSON format like


```
{
   "378466":{
      "url":"COCO_train2014_000000378466.jpg",
      "caption":" A man laying in bed next to his dog. A person that is laying next to a dog. A shirtless man poses
      "words":[
         "metafilter"
      ]
   }
}
```

Step 2 : Making Dictionary from Dataset

Input : Image annotations

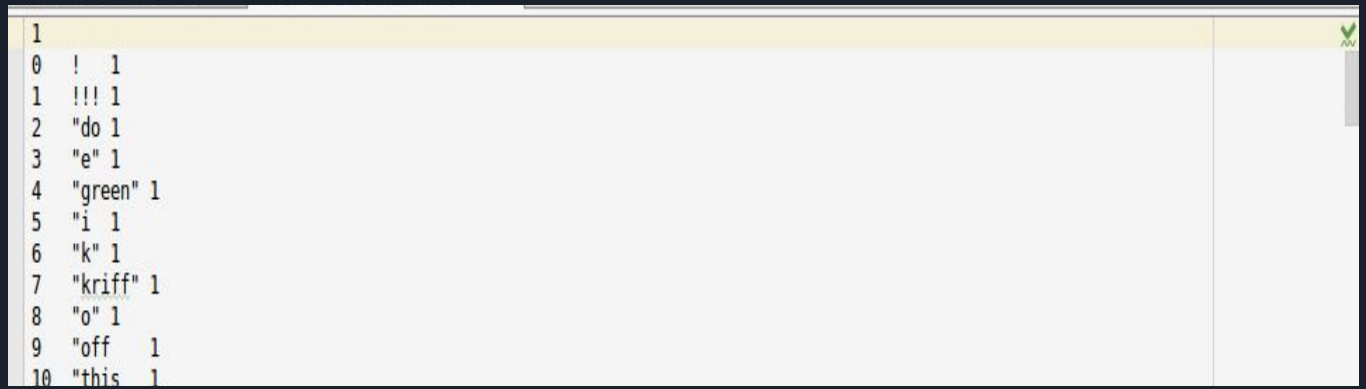
Output : simple list of words


```
["inger", "bruce", "&", "brown", "swis", "heilinger", "laura", "reg.", "photography", "northline", "trinity", "u", "art.com", "yster", "hs-
spl", "24", "bonet", "times", "rent", "pizza", "ranking", "athol", "ave", "700", "500", "rs?", "tors?", "ighbours", "ghbours", "raptors?",
"15", "church", "road", "canyon", "miguel", "house", "oldest", "w", "wilson", "airfrance", "2010", "22.", "03.", "13:21", "aret", "ld",
"olivier", "droits", "caba", "rights", "n", "reserved", "[www.juannonly.com]", "onl", "photo", "juan", "cc", "sa", "pgen", "sans", "dieu",
"copies", "5c", "david", "2008", "salvatori", "silk", "smooth", "delicious", "dark", "dreams", "chocolate"]
```

Step 3 : Convert Dictionary to Gensim Format

Input : dictionary generated in step 2

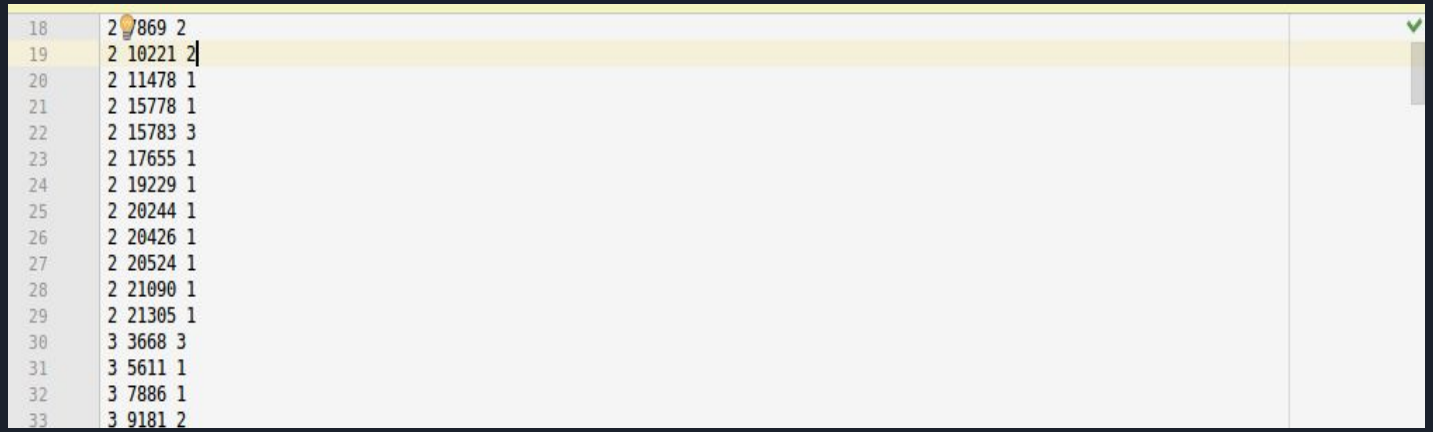
Output : gensim’s class of dictionary which assigns unique id to each word



Step 4 : Preprocess corpus for gensim lda

Input : corpus json file generated in step 1, gensim dictionary generated in step 3

Output : document id - word frequency mapping



Step 5: Learn LDA Topic Modelling

Input : gensim dictionary , bag of words

Output: word distributions over topic (per topic per word probability)

```
INFO:gensim.models.ldamodel:topic diff=0.122713, rho=0.229416
INFO:gensim.models.ldamodel:merging changes from 1686 documents into a model of 43686 documents
INFO:gensim.models.ldamodel:topic #0 (0.100): 0.093*"two" + 0.059*"a" + 0.040*"standing" + 0.037*"field" + 0.025*"men" + 0.022*"flying" + 0.019*"sheep" + 0.016*"giraffes"
 + 0.016*"frisbee" + 0.016*"next"
INFO:gensim.models.ldamodel:topic #7 (0.100): 0.080*"a" + 0.048*"people" + 0.033*"group" + 0.028*"playing" + 0.024*"standing" + 0.022*"street" + 0.018*"game" + 0.017*"tree"
  + 0.016*"frisbee" + 0.014*"wii"
INFO:gensim.models.ldamodel:topic #5 (0.100): 0.117*"a" + 0.041*"man" + 0.032*"people" + 0.028*"snow" + 0.024*"person" + 0.023*"standing" + 0.021*"skateboard" + 0
 .021*"riding" + 0.017*"group" + 0.015*"field"
INFO:gensim.models.ldamodel:topic #6 (0.100): 0.091*"a" + 0.088*"tennis" + 0.041*"room" + 0.033*"ball" + 0.028*"court" + 0.027*"elephant" + 0.026*"living" + 0.023*"two" +
 0.022*"man" + 0.018*"holding"
INFO:gensim.models.ldamodel:topic #1 (0.100): 0.114*"a" + 0.036*"train" + 0.026*"sitting" + 0.022*"laptop" + 0.020*"computer" + 0.019*"desk" + 0.019*"bench" + 0.018*"," +
 0.015*"next" + 0.014*"cake"
```

Step 6 : Generate Word Ranking Results from LDA

Input : Trained LDA Model, Gensim dictionary , Image Annotations and captions

Output : Rankings of words actually present in image

```
['RS? : 18618', 'YOUR : NA', 'TORS? : 13276', 'IGHBOURS : 23486', 'GHBOURS : 10652', 'ARE : NA', 'RAPTORS? : 20157', 'YOUR : NA']
Total words : 24673
----------------------------------------
['15 : 1153', 'Church : 168', 'ROAD : 33', 'THE : NA', 'CANYON : 2970', 'Miguel : 18297', 'HOUSE : 183', 'TO : NA', 'no : 928', 'OLDEST : 13778']
Total words : 24673
----------------------------------------
['W : 672', 'w : 672', 'W : 672', 'Wilson : 2772', 'w : 672']
Total words : 24673
----------------------------------------
['AIRFRANCE : 8487']
Total words : 24673
----------------------------------------
['24 : 2164']
Total words : 24673
----------------------------------------
['2010 : 1919', '22. : 22925', '03. : 24577', '13:21 : 4235']
Total words : 24673
----------------------------------------
['aret : 22222', 'LD : 15591', 'olivier : 7516', 'droits : 20072', 'CABA : 6928']
Total words : 24673
----------------------------------------
```

Step 7 : Generate Labels for training deep CNN

Input : List of images , trained LDA model

Output : Against each image, it stores topic probabilities


```
{"COCO_train2014_000000078077.jpg": ["0", "0", "0", "0.05236661", "0", "0", "0.28067678", "0.24135476", "0", "0.42157415"],
 "COCO_train2014_000000497494.jpg": ["0", "0", "0", "0", "0", "0.17780277", "0.1303499", "0", "0", "0.68666136"],
 "COCO_train2014_000000535795.jpg": ["0", "0", "0.02433229", "0", "0", "0.052881766", "0.36026573", "0.15474012", "0", "0.40407547"],
 "COCO_train2014_000000318857.jpg": ["0", "0", "0", "0", "0", "0.2373438", "0.12718204", "0", "0", "0.6300468"],
 "COCO_train2014_000000480082.jpg": ["0", "0", "0.05262976", "0", "0", "0", "0.35289934", "0.25668213", "0", "0.33327675"],
 "COCO_train2014_000000423964.jpg": ["0", "0", "0.08023027", "0", "0", "0", "0.13454787", "0.3438699", "0", "0.4372133"],
 "COCO_train2014_000000268065.jpg": ["0", "0", "0.02377882", "0", "0", "0.18522625", "0.16028862", "0.101138055", "0", "0.52572125"],
 "COCO_train2014_000000247712.jpg": ["0", "0", "0", "0", "0", "0.22853772", "0.07710342", "0.29863194", "0", "0.39172614"],
```

Step 8 : Fine Tuning Deep CNN

Input : X label : image , Y label : topic probabilities generated in last step, pretrained inception network weights

Output : trained CNN graph

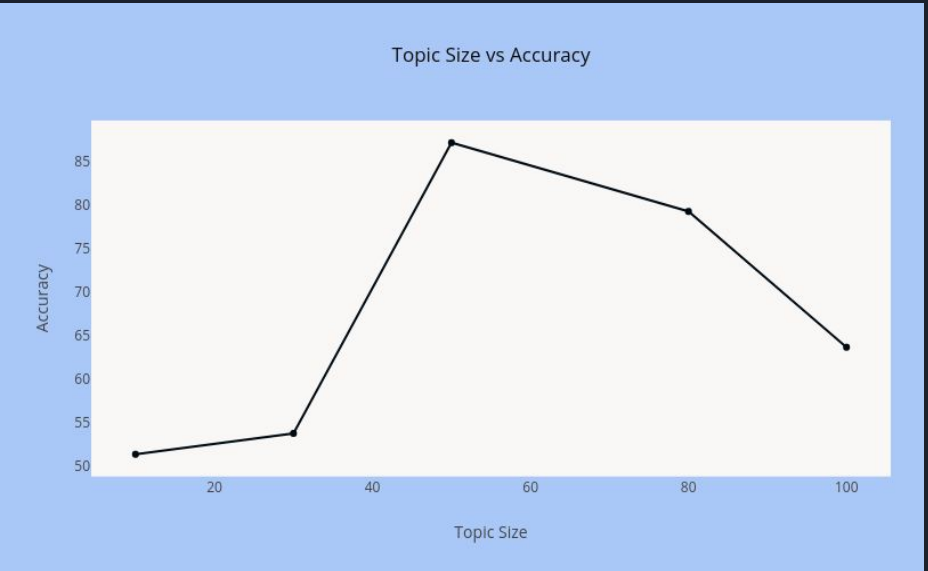
Step 9 : Generate Word Ranking Results from CNN

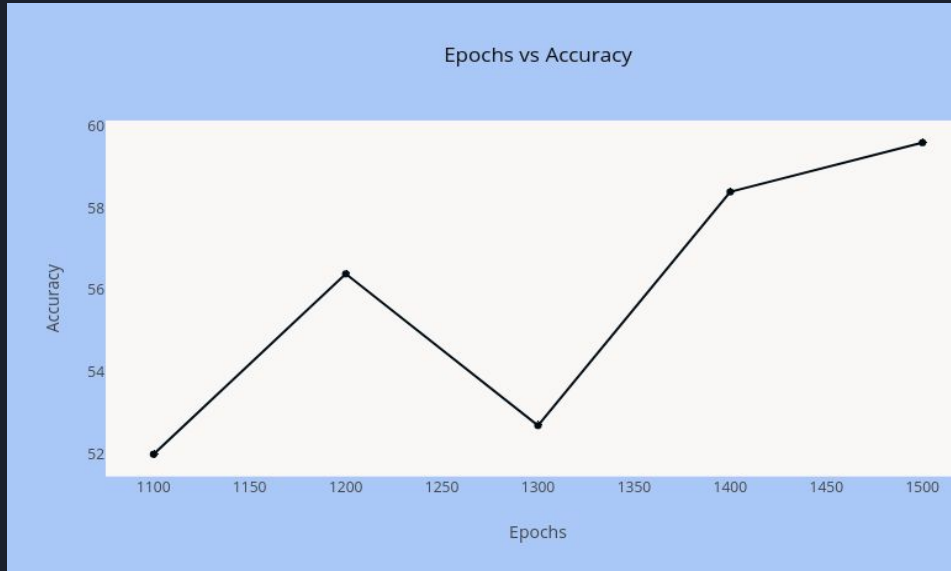
1) Trained CNN generates topic probabilities for unseen image
2) Trained LDA model generates word probabilities , so customized lexicons

```
['AIRFRANCE : 3905']
Image url : COCO_train2014_000000000081.jpg
Total words : 24673
----------------------------------------
['24 : 1931']
Image url : COCO_train2014_000000000086.jpg
Total words : 24673
----------------------------------------
['2010 : 1590', '22. : 22925', '03. : 24577', '13:21 : 6028']
Image url : COCO_train2014_000000000089.jpg
Total words : 24673
----------------------------------------
['Some : NA', 'rights : 3387', 'S : NA', 'by : NA', 'N : 474', 'reserved : 3737', '[www.juannonly.com] : 19054', 'BY : NA', 'Onl : 9298',
Image url : COCO_train2014_000000000109.jpg
Total words : 24673
----------------------------------------
['silk : 4712', 'smooth : 5781', 'delicious : 2276', 'dark : 225', 'and : NA', 'dreams : 5959', 'chocolate : 651']
Image url : COCO_train2014_000000000142.jpg
Total words : 24673
```

Performance

1) For images in the validation set, we compared topic probabilities generated by CNN with that generated by LDA model





LDA for topic modelling

Intuition behind LDA

- doc1. apple banana
- doc2. apple orange
- doc3. banana orange
- doc4. tiger cat
- doc5. tiger dog
- doc6. cat dog

LDA (K = 2)

	Topic 1	Topic2
Apple	33%	0%
Banana	33%	0%
Orange	33%	0%
Tiger	0%	33%
Cat	0%	33%
Dog	0%	33%

	Topic 1	Topic2
doc1	100%	0%
doc2	100%	0%
doc3	100%	0%
doc4	0%	100%
doc5	0%	100%
doc6	0%	100%

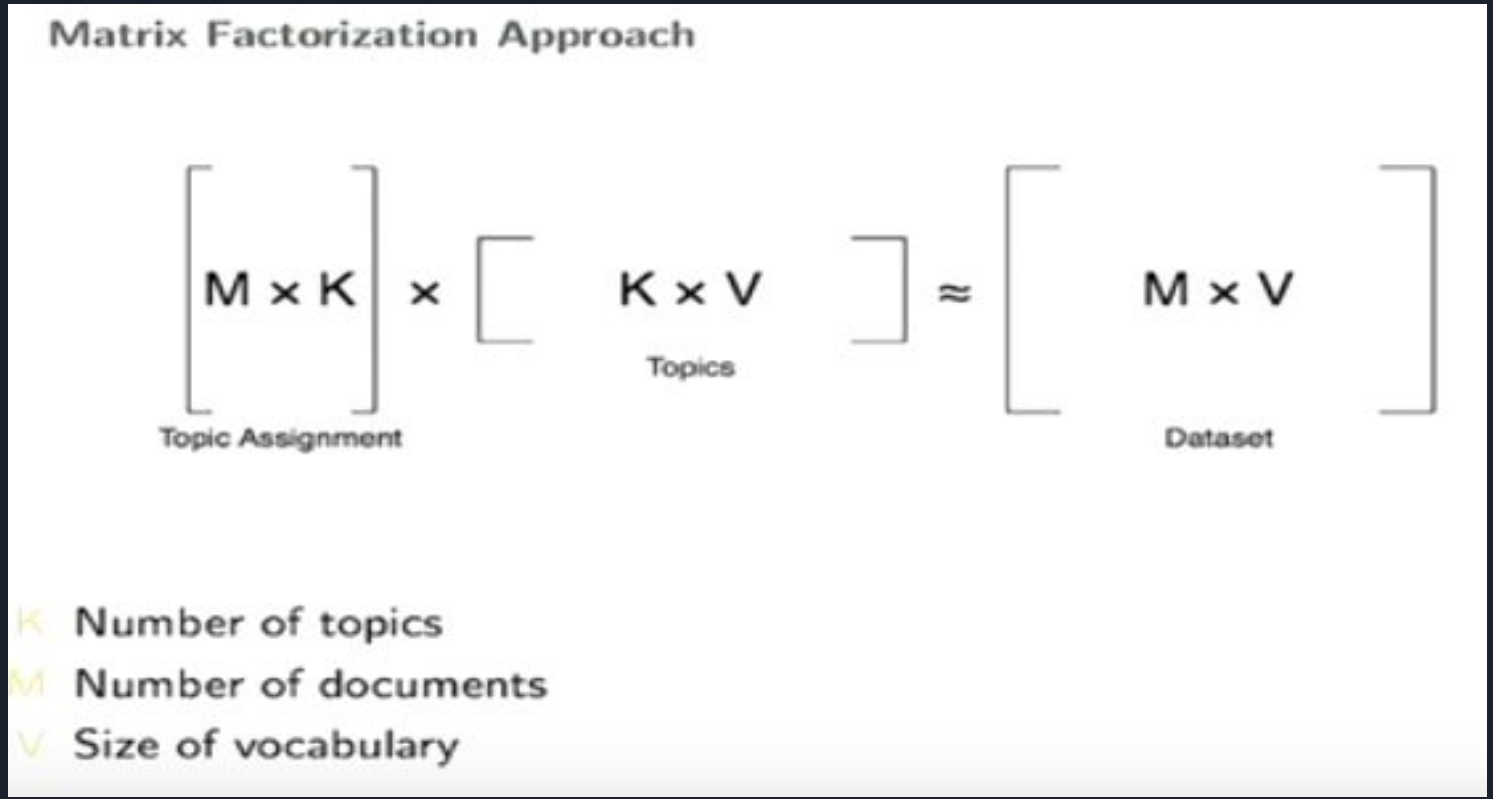
Input (what we have)

Output (what we want)

LDA

1) Unsupervised Learning based on Bag Of Words model
2) Assumes documents are produced from a mixture of topics. Topics then generate words based on their probability distribution.
3) Given a document, LDA backtracks and find topics associated with document and similarly words associated with that topic.

LDA(Continued)

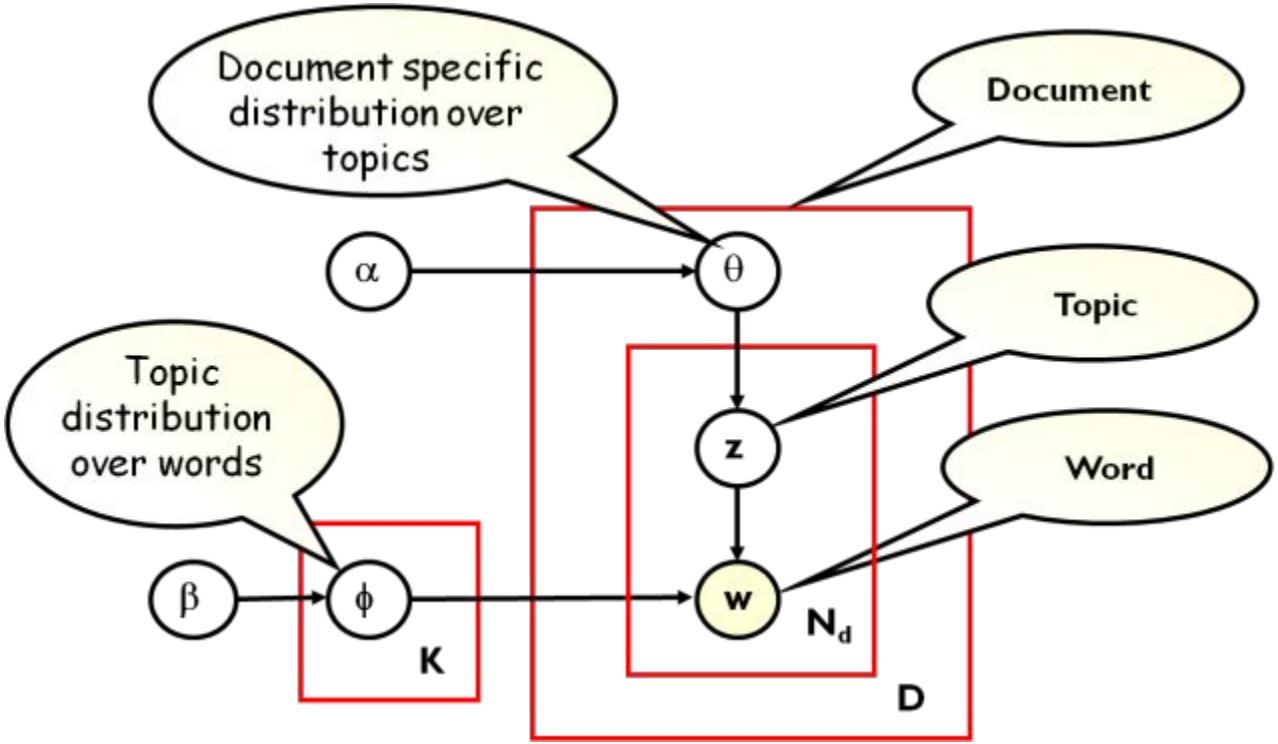


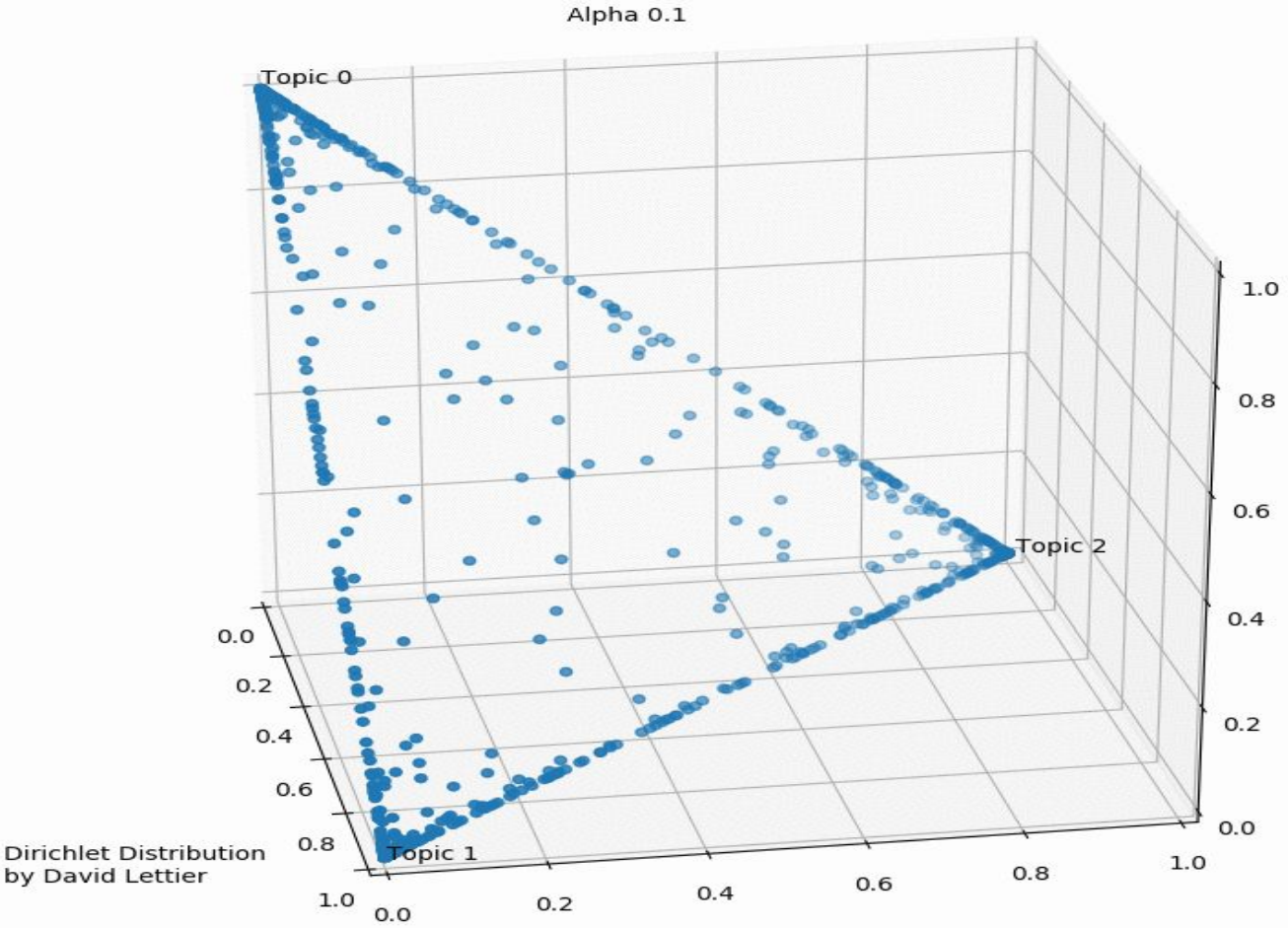
Gibbs Sampling

1) Iterates through each word “w” for each document “d”

2) A new topic “k” is assigned to word “w” with a probability which is a product of two probabilities p1 and p2. P1 - p(topic t / document d) the proportion of words in document d that are currently assigned to topic t. P2 – p(word w / topic t) = the proportion of assignments to topic t over all documents that come from this word w.

LDA Parameters




Alpha 0.1
Topic 0
Topic 1
Topic 2
0.0
0.2
0.4
0.6
0.8
1.0
Dirichlet Distribution
by David Lettier

Generative Process and Formulae

1. Choose $N \sim \text{Poisson}(\xi)$.
2. Choose $\theta \sim \text{Dir}(\alpha)$.
3. For each of the N words w_n:
   (a) Choose a topic $z_n \sim \text{Multinomial}(\theta)$.
   (b) Choose a word w_n from $p(w_n | z_n, \beta)$, a multinomial probability conditioned on the topic z_n.

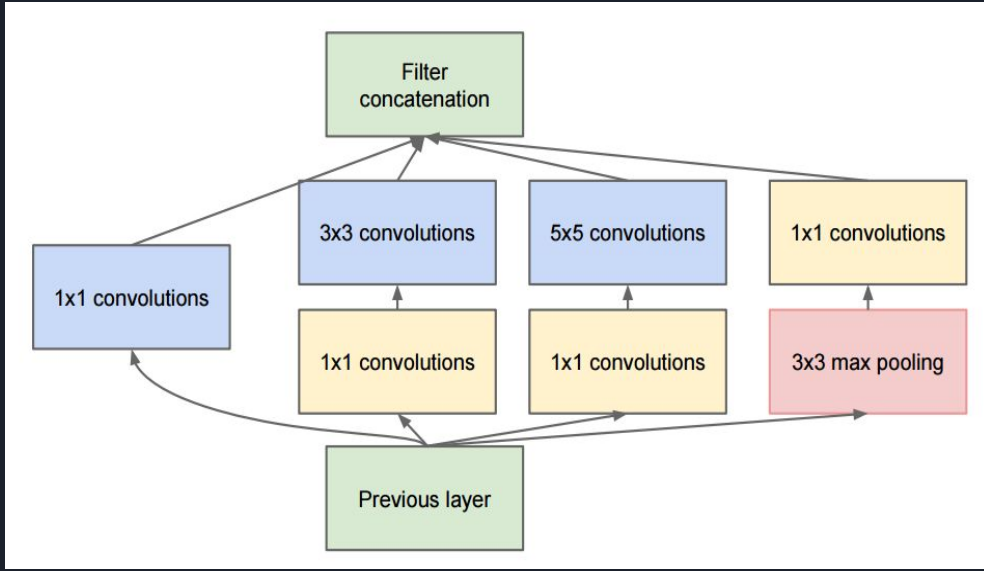
$$P(word \mid text) = \sum_{i=1:K} (P(word \mid topic_i) P(topic_i \mid text))$$

$$P(word \mid image) = \sum_{i=1:K} (P(word \mid topic_i) P(topic_i \mid image))$$

Let's say you have input data x and you want to classify the data into labels y. A generative model learns the **joint** probability distribution p(x,y) and a discriminative model learns the **conditional** probability distribution p(y|x).

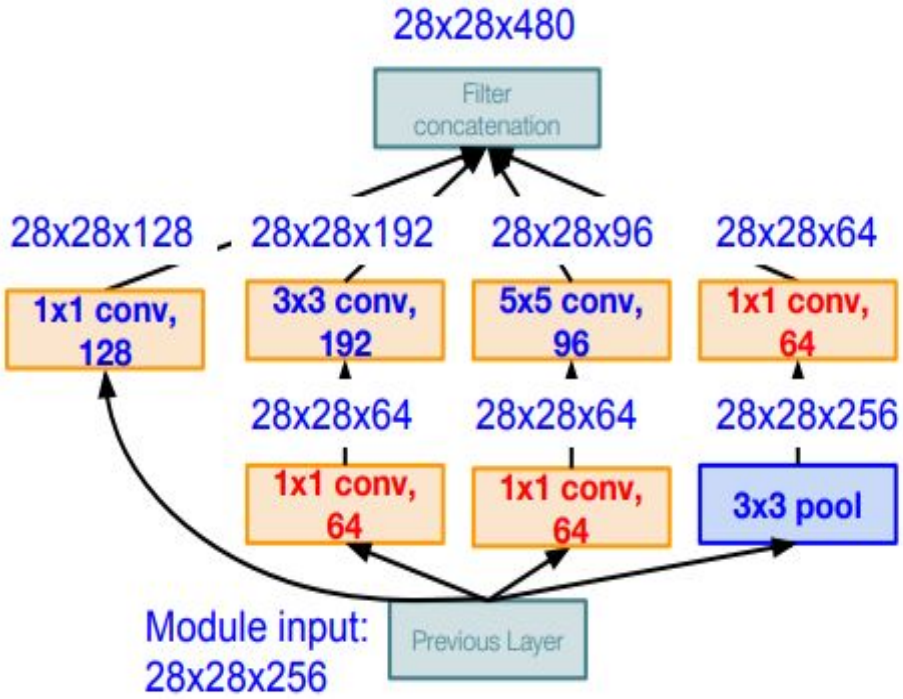
Inception Network

- Inception module: design a good local network topology (network within a network) and then stack these modules on top of each other
- Apply parallel filter operations on the input from previous layer: - Multiple receptive field sizes for convolution (1x1, 3x3, 5x5) - Pooling operation (3x3) Concatenate all filter outputs together depth-wise
- Pooling layer also preserves feature depth, which means total depth after concatenation can only grow at every layer!
- 1X1 Conv preserves spatial dimensions, reduces depth! Projects depth to lower dimension (combination of feature maps)
- 22 Layers



Case Study: GoogLeNet

[Szegedy et al., 2014]



Inception module with dimension reduction

Using same parallel layers as naive example, and adding "1x1 conv, 64 filter" bottlenecks:

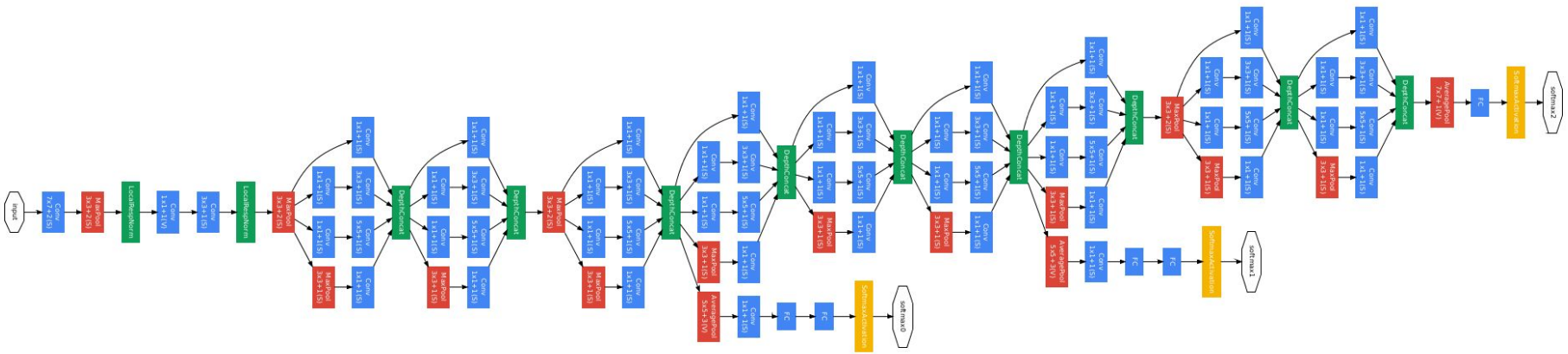
Conv Ops:

[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 64] 28x28x64x1x1x256
[1x1 conv, 128] 28x28x128x1x1x256
[3x3 conv, 192] 28x28x192x3x3x64
[5x5 conv, 96] 28x28x96x5x5x64
[1x1 conv, 64] 28x28x64x1x1x256

Total: 358M ops

Compared to 854M ops for naive version
Bottleneck can also reduce depth after pooling layer

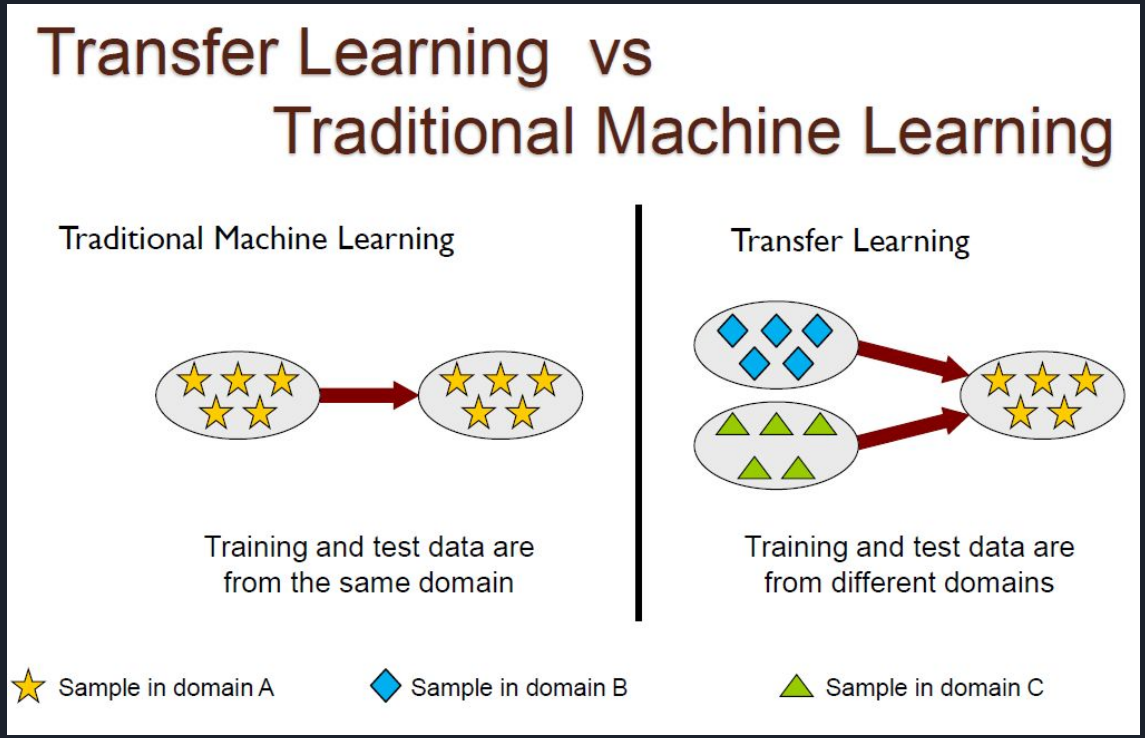
Inception Network(Continued)



Given large depth, ability to backprop is a concern. Layers in the middle of the networks should be very discriminative. Add auxiliary classifiers. Loss weighed by 0.3 in aggregate loss

Transfer Learning

- The ability of a system to recognize and apply knowledge and skills learned in previous tasks to novel tasks (or new domains)
- Only the final fully connected layer modified to accommodate the number of topics in current experiment.



Application Ideas

1. Context based image similarity detection.
2. Generate images based on a text description.
3. Sentiment detection from image.