Paragraph Topic Extraction
- From Naive Bayes to Convolutional Neural Network -
Edward Ng, Eugene Nho
Stanford University Dept. of Electrical Engineering, Stanford Graduate School of Business
{edjng, enho}@stanford.edu

Motivation & Problem Definition

- “Exponentially” increasing information vs constant human ability to ingest text. Automatic topic extraction can help close this gap
- Example use cases: detecting relevant sections from SEC filings, market reports, legal documents, etc.

Formalized Problem: Given a paragraph of text as input, predict what topics the input is about, from a given set of topics (multilabel)

Overall Approach: Data, Features, and Model

Data
- Wikipedia “abstract” (first paragraph) as input (x), their category assignments as labels (y)
- Total 500K inputs, labeled as one or more of (math, politics, computer science, film, music) (multilabel)
- Wikipedia chosen because of readily available human-tagged labels

Features
- Term Frequency-Inverse Document Frequency (tf-idf): Common text normalization technique
- GloVe: Richer representation; captures co-occurrence of words, which helps topic detection

Model
- Naive Bayes [tf-idf]
- One-vs-rest classifier (OvR) [GloVe]
- Latent Dirichlet Allocation (LDA) + OvR [tf]
- LDA + GloVe + OvR [tf]
- Convolutional Neural Network (CNN) [GloVe]

Overall Approach: Schematic

Overall Approach: Data, Features, and Model

Results and Discussion

- As we utilized richer representations of input text, performance (F1 score) increased overall
- Power of latent topic representation (LDA) is notable, with ~1.6x increase in F1 score from NB (vs ~1.25x GloVe)
- Topics captured by LDA closely mirror original labels, e.g. “Topic 3” represented by words {album, song, released, single, ...}, “Topic 4” by {theory, displaystyle, soviet, university, graph analysis, mathematical, ...}
- CNN provided superior results despite its primary use case being image classification
- Future work: (1) experimenting with different hyperparameter settings for CNN

F1 score of various approaches

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>NB</td>
<td>76</td>
<td>48</td>
</tr>
<tr>
<td>OvR + GloVe</td>
<td>62</td>
<td>78</td>
</tr>
<tr>
<td>LDA</td>
<td>82</td>
<td>66</td>
</tr>
<tr>
<td>LDA + GloVe + OvR</td>
<td>92</td>
<td>84</td>
</tr>
<tr>
<td>CNN</td>
<td>88</td>
<td></td>
</tr>
</tbody>
</table>

1 sklearn, numpy, tensorflow, and keras were used in our development. Our classifiers assumed the default hyperparameters provided by the respective libraries

We focused on first building a “quick-and-dirty” baseline, then added more sophisticated features / models and observed how performance improved:

1: Common baseline model for text classification
2 One-vs-rest supports multilabel learning; richer feature (GloVe)
3 To capture latent topics more effectively
4 Complementary: GloVe (local focus) and LDA (more global)
5 Fast; generalizes well; local word order is not important