Title Classification for Cybersecurity
People: Brian Jones (Manager), Zachary Abzug (Mentor), Aaron Liu (Co-intern)

Introduction

- Proofpoint is a cybersecurity company providing software as a service
- Clients interested in how their people are being attacked. Insight on groups of employees more actionable than at the individual level
- The Title Classifier seeks to classify an employee’s title and department into a normalized business function and seniority

Methods

- Problem: Supervised multi-class classification problem in natural language processing
- Solution: Long short-term memory (LSTM) neural network (right)
- Extension of recurrent neural network
- Designed for sequential / variable length input (text)
- Address vanishing gradient found in vanilla recurrent neural networks
- Cell state (top line) represents long-term memory while the hidden state (bottom line) interacts with input for updates
- Final hidden state used for classification

High-level flowchart of classifier

1. Batch of title-department pairs obtained from the training data
2. Tokenization of input at character-level
   - Word and subword level also considered
3. Tokens converted into real-valued vectors (embeddings) to be processed by network
4. Embeddings sequentially fed into LSTM layer, updating hidden and cell states at each step
5. Final set of dense layers used to bring output to appropriate dimensionality

- Cross-entropy loss function used:
  \[ \text{loss}(x, \text{class}) = - \log \left( \frac{e^{x_i}}{\sum_j e^{x_j}} \right) = -x_{\text{class}} + \log \left( \sum_j e^{x[j]} \right) \]

- \( x \in \mathbb{R}^n \) is the unnamed output of the final dense layer
- Convert to distribution via inner product

Application

- 3463 hand-labeled examples
- Problems with Data
  - Missing departments (13%)
  - Foreign languages
  - Acronyms
- Cross-functional mapping
- Trained over 5 epochs until validation loss stopped decreasing

<table>
<thead>
<tr>
<th>Generic Business Function</th>
<th>Seniority</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administration</td>
<td>Executive</td>
</tr>
<tr>
<td>Product Dev / Services</td>
<td>VP</td>
</tr>
<tr>
<td>Bizdev / Strategic</td>
<td>Manager / Director</td>
</tr>
<tr>
<td>Supply Chain</td>
<td>Employee</td>
</tr>
<tr>
<td>Finance</td>
<td>Part-time Employee</td>
</tr>
<tr>
<td>Purchasing</td>
<td>Unsure</td>
</tr>
<tr>
<td>HR</td>
<td></td>
</tr>
<tr>
<td>Sales</td>
<td></td>
</tr>
<tr>
<td>IT</td>
<td></td>
</tr>
<tr>
<td>Facilities</td>
<td></td>
</tr>
<tr>
<td>Customer Service</td>
<td></td>
</tr>
<tr>
<td>Marketing</td>
<td></td>
</tr>
</tbody>
</table>

Latent Dirichlet Allocation with Collapsed Gibbs Sampling (Porteous et al., 2008[1])
Collaborators: Pierre Gardan

Introduction

- Latent Dirichlet Allocation (LDA) is a generative, probabilistic model for discrete data (text). We assume that words are drawn according to document specific mixture distributions of topics. The mixture distributions of words per topic and topic per document are described via Dirichlet distributions
- Primary use case explored is corpus summarization: how can we describe the underlying topics in a collection of documents?
- The exact posterior is difficult to obtain samples from. We implement a collapsed Gibbs sampler to obtain a chain of latent topics, then use these to obtain estimates of the remaining parameters

Methods

- A topic is simply a distribution over words
- Model each document as a mixture over K topics
- Model contents of document as multinomial over W word vocabulary
- Generative process for word \( w_j \)
  1. Draw document-specific topic mixtures \( \theta_i \sim \text{Dirichlet}(\alpha) \)
  2. Draw topic-specific word mixtures \( \phi_k \sim \text{Dirichlet}(\beta) \)
  3. Draw latent topic \( z_{ij} \sim \text{Categorical}(\theta_i) \)
  4. Draw word given topic \( w_j \sim \text{Categorical}(\phi_{z_{ij}}) \)

- Marginal joint distribution of \( z_{kj} \) obtained by integrating theta and phi over full joint distribution:
  \[
  p(z_{ij} = k | x, w) = \frac{1}{Z} \sum_k p(z_{ij} = k | x, w, \alpha, \beta) = \frac{1}{Z} \sum_k p(z_{ij} = k | x) + \frac{\alpha_k}{N_k} + \frac{\beta_k}{N_k + W} \]

- Use to construct collapsed Gibbs sampler on topic values alone
- \( N_k \) : Number of times word \( w \) is assigned to topic \( k \) in document \( j \)
  \[
  p(z_{ij} = k | x, w, \alpha, \beta) = \frac{1}{Z} \sum_k p(z_{ij} = k | x) + \frac{\alpha_k}{N_k} + \frac{\beta_k}{N_k + W} \]

- Recover posterior estimates of other parameters via:
  \[
  \hat{\theta}_{kj} = \frac{N_{kj} + \alpha}{N_j + K \alpha} \quad \hat{\phi}_{kj} = \frac{N_{kj} + \beta}{N_k + W \beta} \]

Application

- Pre-processing
  - Tokenization
  - Lowercasing
  - Removal of stop words
  - Lemmatization
- Datasets
  - 20 Newsgroups
    - \( D = 18000 \) documents
    - \( k = 20 \) topics
  - Reuters-21578 (subset)
    - \( D = 10000 \) documents
    - \( k = 5 \) topics

<table>
<thead>
<tr>
<th>Interpreted Topic</th>
<th>Matched Topic</th>
<th>Top 5 Words by Posterior Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computers / Cryptography</td>
<td>Sci.crypt</td>
<td>(<code>key</code>, 0.056), (chip', 0.023), (encryption', 0.020), (bit', 0.015), (system', 0.015)</td>
</tr>
<tr>
<td>Sports</td>
<td>Rec.sport</td>
<td>(game', 0.043), (team', 0.033), (year', 0.027), (play', 0.020), (algorithm', 0.021)</td>
</tr>
<tr>
<td>Religion</td>
<td>Soc.religion</td>
<td>(god', 0.0497), (jesus', 0.0219), (bible', 0.0122), (life', 0.0128)</td>
</tr>
</tbody>
</table>

References: