Beyond Word Embeddings: Dense Representations for Multi-Modal Data.

Luis Armona  larmona@stanford.edu
Ralph Edezhath  redezhath@chegg.com
José González-Brenes  jgonzalez@chegg.com
Introduction
Informally, a lower-dimensional projection of a high-dimensional vector

Usually, embeddings have a denser (less sparse) representation of the original vector

- Example: embeddings for bag-of-words

  BOW vector for the word “zone”   Embedding

<table>
<thead>
<tr>
<th>a</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>an</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>zone</td>
<td>1</td>
</tr>
<tr>
<td>zoon</td>
<td>0</td>
</tr>
<tr>
<td>zouck</td>
<td>0</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.3</td>
<td></td>
</tr>
<tr>
<td>-1</td>
<td></td>
</tr>
<tr>
<td>2.5</td>
<td></td>
</tr>
</tbody>
</table>
Supervised Embeddings

- Embeddings are tuned for the prediction task that you care about
- Canonical example: Netflix challenge
  - Bag of movies -> Embedding of movies
  - Bag of users -> embedding of users
  - The dot product of the embeddings predict how much a user likes a movie
Self-supervised Embeddings

• No explicit labels provided for training
• The model learns the structure of the data, hoping that it will be useful for a downstream task
• Example:
  • Word2Vec learns the structure of text based on context
  • used for classification, sentiment analysis, etc.
Not all data is text!
### Multi-modal data – Example

<table>
<thead>
<tr>
<th>Lead actor/actress</th>
<th>Movie Name</th>
<th>Length</th>
<th>Synopsis</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nicholas Cage</td>
<td>Ghost Rider</td>
<td>95 mins</td>
<td>Motorcycle stuntman sold his soul..</td>
<td>Color</td>
</tr>
<tr>
<td>Humphrey Bogart</td>
<td>Casablanca</td>
<td>102 mins</td>
<td>A famed rebel, and with Germans on his tail...</td>
<td>B&amp;W</td>
</tr>
<tr>
<td>Warren Dudley</td>
<td>Cage</td>
<td>80 mins</td>
<td>A GI in the Vietnam War saves ...</td>
<td>Color</td>
</tr>
</tbody>
</table>

- **Multi-modal:** Text, categorical, real, pictures ...
- How would you learn self-supervised embeddings using this metadata? Two options with existing technology
Existing technology: Option 1

• Build pseudo-documents and use one of the Word2Vec algorithms to find embeddings:

- Nicholas Cage Ghost Rider
  95
  Motorcycle ...
  color

- Humphrey Bogart
  Casablanca 102
  ...B&W

- Warren Dudley Cage 80 ...
  color
Problems

In general, this approach doesn’t know the types of the entities (actor, title, length, ...).

- Nicholas Cage Ghost Rider 95 Motorcycle ...
- Humphrey Bogart Casablanca 102 ...
- Warren Dudley Cage 80 ...

It cannot exploit type information (95 is closer to 80, than to 195)
Can’t work around this

• Some trivial work-around have been proposed
  • e.g. using a prefix for tokens to indicate type
  • (Nicholas Cage → actor_Nicholas_Cage)

• In general, they don’t work
  • Order lacks meaning
  • Still no structure on type similarities
  • Metadata features sampled according to frequency in ENTIRE vocabulary of pseudo-documents
Option 2: Build an ad hoc model specific to multi-modal setting. There are many models to calculate self-supervised embeddings for a specific use case. Examples: Doc2Vec (Le and Mikolov 2014), Tagspace (Weston et al. 2014)

This is very time consuming, and it requires a lot of work
Feat2Vec: Embedding features, not just text!
Objective

• Our purpose is to present a general-purpose model that is able to embed data with multiple types of features (e.g. categories, text, images, numerical)
• Support both supervised and self-supervised embeddings
Network Architecture

Multi-modal Data

\[ d_1 = 50k \]

\[ X_1 \rightarrow \phi_1 \]

\[ d_2 = 1 \]

\[ X_2 \rightarrow \phi_2 \]

\[ \ldots \]

\[ d_\phi = 50 \]

\[ \phi_i \cdot \phi_j \]

\[ \sum_i \sum_{j \geq i} \phi_i \cdot \phi_j \rightarrow \hat{y} \]
How does Feat2Vec create negative samples?

• Choose a row in the training data randomly
• First choose a feature (column) randomly
• We sample proportionally to how complex is a feature
  • (i.e., how many parameters we need to learn from the feature)
• Substitute the feature value proportional to the empirical distribution
• Equivalent to ensemble of supervised classifiers for each feature
• Math and details in the paper
# Feat2Vec vs Word2Vec

<table>
<thead>
<tr>
<th></th>
<th>Word2Vec</th>
<th>Feat2Vec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Predictive model</td>
<td>Binary classifier</td>
<td></td>
</tr>
<tr>
<td>Positive examples</td>
<td>Training data</td>
<td></td>
</tr>
<tr>
<td>Negative examples</td>
<td>Word combinations that are unseen</td>
<td>Feature combinations that are unseen</td>
</tr>
<tr>
<td>How are negative examples generated?</td>
<td>Substituting a word randomly</td>
<td>Substituting a feature value randomly</td>
</tr>
<tr>
<td>Loss</td>
<td>Negative Sampling (logistic loss)</td>
<td>Noise contrastive estimation loss</td>
</tr>
</tbody>
</table>
Empirical Results
Yelp Reviews

• Use Yelp Review Dataset of 4.7M restaurant reviews. Groups:
  – Feature Groups: User ID, Restaurant ID, Review Text, Funny Indicator, Rating
  – Use CNN for Review Text
• Compare to Word2Vec (CBOW, Tokenized features) and Doc2Vec (DM).
• Evaluation: Train 50d Embeddings on 90% of data, then predict ratings of 10% held-out sample using review text embedding (CNN for F2V, average of word vectors for D2V/W2V)
• Run similar experiments on IMDB movie dataset + anonymized education data.
# Results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word2Vec</td>
<td>61.3%</td>
</tr>
<tr>
<td>Doc2Vec</td>
<td>73%</td>
</tr>
<tr>
<td>Feat2Vec</td>
<td>55%</td>
</tr>
</tbody>
</table>

- Feat2Vec = 10% improvement over Word2Vec, 30% over Doc2Vec.
Conclusion
Conclusion

• Feat2Vec uses information about feature types to learn better embeddings
• First self-supervised embedding method accommodating multiple feature types.
• We also implement Feat2Vec in supervised setting and show it outperforms state-of-the-art results on Yelp
• Codebase available on github (CheggEng) for future consumption
Thank You!!