Improving Search for ExploreCourses
Sam Redmond, Eddie Wang

Overview
The existing implementation of course search for ExploreCourses relies on naïve substring matching. We build an improved search model for the Stanford course dataset by composing document vector and word vector models.

Objective: Improve relevance of course search results

Dataset and Processing
2015-2016 course listings from ExploreCourses
• 14K courses (code, title, description)
• 37K unique words
• ~1.3M tokens

We preprocess each course
• Split sentences with NLTK
• Parse tokens with Penn-Treebank tokenization
• Compute and condense the most likely bigrams
Each sentence is labelled with the course code and a unique identifier for training.
For example:
Title: machine learning
Description: topics: statistical pattern recognition, linear ...
becomes
[{'machine_learning'}, {'topics', ':', 'statistical', 'pattern_recognition', ':', 'linear', ...}, ...], ['cs229']

Methods

Training
Document model
• Obtain document and word vectors by training a distributed memory (PV-DM) paragraph2vec model
• Course descriptions are mapped to dense, fixed-length vectors
• Word vectors initialized to pre-trained values (randomly if unseen)
• Pre-trained tokens learn at 1/5 ordinary learning rate
• Repeatedly train until the word and label vectors reach convergence.

Word model
• Each word maps to its most similar courses
• For a given word, each course is scored by: \( \frac{f_{\text{course}}}{f_{\text{word}} + f_{\text{total}}} \)
  similar to the relative frequency metric used in Naïve Bayes text

Suggesting
Document Model
Query
Find similar labels
Extract courses
Word Model
Find similar words
Lookup associated courses
Composite Predictions

Querying
• Convert query string to query vector using trained paragraph2vec model
• Document model: find closest label(s) by cosine similarity, then extract course codes from label(s)
• Word model: find similar words to the query vector, then look up associated courses
• Average results together, weighted by confidence, to produce final result

Evaluation and Results

Qualitative
• On average, our composite model outperforms ExploreCourses.
• Worst case: not worse than ExploreCourses

Sample queries for which the difference is especially apparent

<table>
<thead>
<tr>
<th>Query</th>
<th>ExploreCourses</th>
<th>Composite Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer Programming</td>
<td>Biophysics of Multi-cellular Systems</td>
<td>Introduction to Computing Principles</td>
</tr>
<tr>
<td>Physics</td>
<td>Programming Service Project</td>
<td>Programming Service Project</td>
</tr>
<tr>
<td>Violin Lessons</td>
<td>None</td>
<td>Introductory Violin Class</td>
</tr>
<tr>
<td>Linear Algebra</td>
<td>Advanced Feedback Control Design</td>
<td>Violin Advanced Violin</td>
</tr>
<tr>
<td>Linear Algebra</td>
<td>Design</td>
<td>Modern Algebra</td>
</tr>
<tr>
<td>Linear Algebra and Differential Calculus</td>
<td>Mechanical Vibrations</td>
<td>Linear Algebra and Differential Calculus</td>
</tr>
</tbody>
</table>

Top three search results for the ExploreCourses model and our composite model run against three sample queries.

Quantitative
• Asked 20 students to create ten queries within several categories and score the results out of 5.
• Other user interaction metrics are withheld for privacy reasons

<table>
<thead>
<tr>
<th>Category</th>
<th>ExploreCourses</th>
<th>Paragraph2Vec</th>
<th>Word2Vec</th>
<th>Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find a specific course</td>
<td>3.4</td>
<td>2.5</td>
<td>4.2</td>
<td>4.4</td>
</tr>
<tr>
<td>Find courses on a specific topic</td>
<td>2.2</td>
<td>3.1</td>
<td>3.8</td>
<td>4.0</td>
</tr>
<tr>
<td>Find courses on a general topic</td>
<td>1.3</td>
<td>3.6</td>
<td>3.2</td>
<td>3.9</td>
</tr>
<tr>
<td>Total</td>
<td>2.3</td>
<td>3.1</td>
<td>3.7</td>
<td>4.1</td>
</tr>
</tbody>
</table>

Average scores (out of 5) given by n=20 students assessing performance of different models across a variety of search categories