TensorBoard: Embedding Visualization

Embeddings are ubiquitous in machine learning, appearing in recommender systems, NLP, and many other applications. Indeed, in the context of TensorFlow, it's natural to view tensors (or slices of tensors) as points in space, so almost any TensorFlow system will naturally give rise to various embeddings.

TensorBoard has a built-in visualizer, called the *Embedding Projector*, for interactive visualization and analysis of high-dimensional data like embeddings. The embedding projector will read the embeddings from your model checkpoint file. Although it's most useful for embeddings, it will load any 2D tensor, including your training weights.

To learn more about embeddings and how to train them, see the Vector Representations of Words (https://www.tensorflow.org/tutorials/word2vec) tutorial. If you are interested in embeddings of images, check out this article (http://colah.github.io/posts/2014-10-Visualizing-MNIST/) for interesting visualizations of MNIST images. On the other hand, if you are interested in word embeddings, this article (http://colah.github.io/posts/2015-01-Visualizing-Representations/) gives a good introduction.

By default, the Embedding Projector projects the high-dimensional data into 3 dimensions using principal component analysis (https://en.wikipedia.org/wiki/Principal_component_analysis). For a visual explanation of PCA, see this article (http://setosa.io/ev/principal-component-analysis/). Another very useful projection you can use is t-SNE (https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding). We talk about more t-SNE later in the tutorial.

If you are working with an embedding, you'll probably want to attach labels/images to the data points. You can do this by generating a metadata file (#metadata) containing the labels for each point and configuring the projector either by using our Python API, or manually constructing and saving a **`projector_config.pbtxt`**

(https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/tensorboard/plugins/projector/projector_config.proto) in the same directory as your checkpoint file.

Setup

For in depth information on how to run TensorBoard and make sure you are logging all the necessary information, see TensorBoard: Visualizing Learning (https://www.tensorflow.org/get_started/summaries_and_tensorboard).

To visualize your embeddings, there are 3 things you need to do:

1) Setup a 2D tensor that holds your embedding(s).

```
embedding_var = tf.Variable(....)
```

2) Periodically save your model variables in a checkpoint in **`LOG_DIR`**.

```
saver = tf.train.Saver()
saver.save(session, os.path.join(LOG_DIR, "model.ckpt"), step)
```

3) (Optional) Associate metadata with your embedding.

If you have any metadata (labels, images) associated with your embedding, you can tell TensorBoard about it either by directly storing a **`projector_config.pbtxt`** (https://github.com/tensorflow/tensorflow/blob/master/tensorflow/contrib/tensorboard/plugins/projector/projector_config.proto) in the **`LOG_DIR`**, or use our python API.

For instance, the following **`projector_config.ptxt`** associates the **`word_embedding`** tensor with metadata stored in **`$LOG_DIR/metadata.tsv`**:

```
embeddings {
  tensor_name: 'word_embedding'
  metadata_path: '$LOG_DIR/metadata.tsv'
}
```

The same config can be produced programmatically using the following code snippet:

```
from tensorflow.contrib.tensorboard.plugins import projector

# Create randomly initialized embedding weights which will be trained.
N = 10000 # Number of items (vocab size).
D = 200 # Dimensionality of the embedding.
embedding_var = tf.Variable(tf.random_normal([N,D]), name='word_embedding')

# Format: tensorflow/contrib/tensorboard/plugins/projector/projector_config.proto
config = projector.ProjectorConfig()

# You can add multiple embeddings. Here we add only one.
embedding = config.embeddings.add()
embedding.tensor_name = embedding_var.name
# Link this tensor to its metadata file (e.g. labels).
embedding.metadata_path = os.path.join(LOG_DIR, 'metadata.tsv')

# Use the same LOG_DIR where you stored your checkpoint.
summary_writer = tf.summary.FileWriter(LOG_DIR)

# The next line writes a projector_config.pbtxt in the LOG_DIR. TensorBoard will
# read this file during startup.
projector.visualize_embeddings(summary_writer, config)
```

After running your model and training your embeddings, run TensorBoard and point it to the **LOG_DIR** of the job.

```
tensorboard --logdir=LOG_DIR
```

Then click on the *Embeddings* tab on the top pane and select the appropriate run (if there are more than one run).

Metadata

Usually embeddings have metadata associated with it (e.g. labels, images). The metadata should be stored in a separate file outside of the model checkpoint since the metadata is not a trainable parameter of the model. The format should be a TSV file (https://en.wikipedia.org/wiki/Tab-separated_values) (tab characters shown in red) with the first line containing column headers (shown in bold) and subsequent lines contain the metadata values:

```
Word\tFrequency
Airplane\t345
Car\t241
...
```

There is no explicit key shared with the main data file; instead, the order in the metadata file is assumed to match the order in the embedding tensor. In other words, the first line is the header information and the (i+1)-th line in the metadata file corresponds to the i-th row of the embedding tensor stored in the checkpoint.

Note: If the TSV metadata file has only a single column, then we don't expect a header row, and assume each row is the label of the embedding. We include this exception because it matches the commonly-used "vocab file" format.

Images

If you have images associated with your embeddings, you will need to produce a single image consisting of small thumbnails of

each data point. This is known as the sprite image (https://www.google.com/webhp#q=what+is+a+sprite+image). The sprite should have the same number of rows and columns with thumbnails stored in row-first order: the first data point placed in the top left and the last data point in the bottom right:

0	1	2
3	4	5
6	7	

Note in the example above that the last row doesn't have to be filled. For a concrete example of a sprite, see this sprite image (https://www.tensorflow.org/images/mnist_10k_sprite.png) of 10,000 MNIST digits (100x100).

Note: We currently support sprites up to 8192px X 8192px.

After constructing the sprite, you need to tell the Embedding Projector where to find it:

```
embedding.sprite.image_path = PATH_TO_SPRITE_IMAGE
# Specify the width and height of a single thumbnail.
embedding.sprite.single_image_dim.extend([w, h])
```

Interaction

The Embedding Projector has three panels:

1. *Data panel* on the top left, where you can choose the run, the embedding tensor and data columns to color and label points by.
2. *Projections panel* on the bottom left, where you choose the type of projection (e.g. PCA, t-SNE).

3. *Inspector panel* on the right side, where you can search for particular points and see a list of nearest neighbors.

Projections

The Embedding Projector has three methods of reducing the dimensionality of a data set: two linear and one nonlinear. Each method can be used to create either a two- or three-dimensional view.

Principal Component Analysis A straightforward technique for reducing dimensions is Principal Component Analysis (PCA). The Embedding Projector computes the top 10 principal components. The menu lets you project those components onto any combination of two or three. PCA is a linear projection, often effective at examining global geometry.

t-SNE A popular non-linear dimensionality reduction technique is t-SNE. The Embedding Projector offers both two- and three-dimensional t-SNE views. Layout is performed client-side animating every step of the algorithm. Because t-SNE often preserves some local structure, it is useful for exploring local neighborhoods and finding clusters. Although extremely useful for visualizing high-dimensional data, t-SNE plots can sometimes be mysterious or misleading. See this great article (http://distill.pub/2016/misread-tsne/) for how to use t-SNE effectively.

Custom You can also construct specialized linear projections based on text searches for finding meaningful directions in space. To define a projection axis, enter two search strings or regular expressions. The program computes the centroids of the sets of points whose labels match these searches, and uses the difference vector between centroids as a projection axis.

Navigation

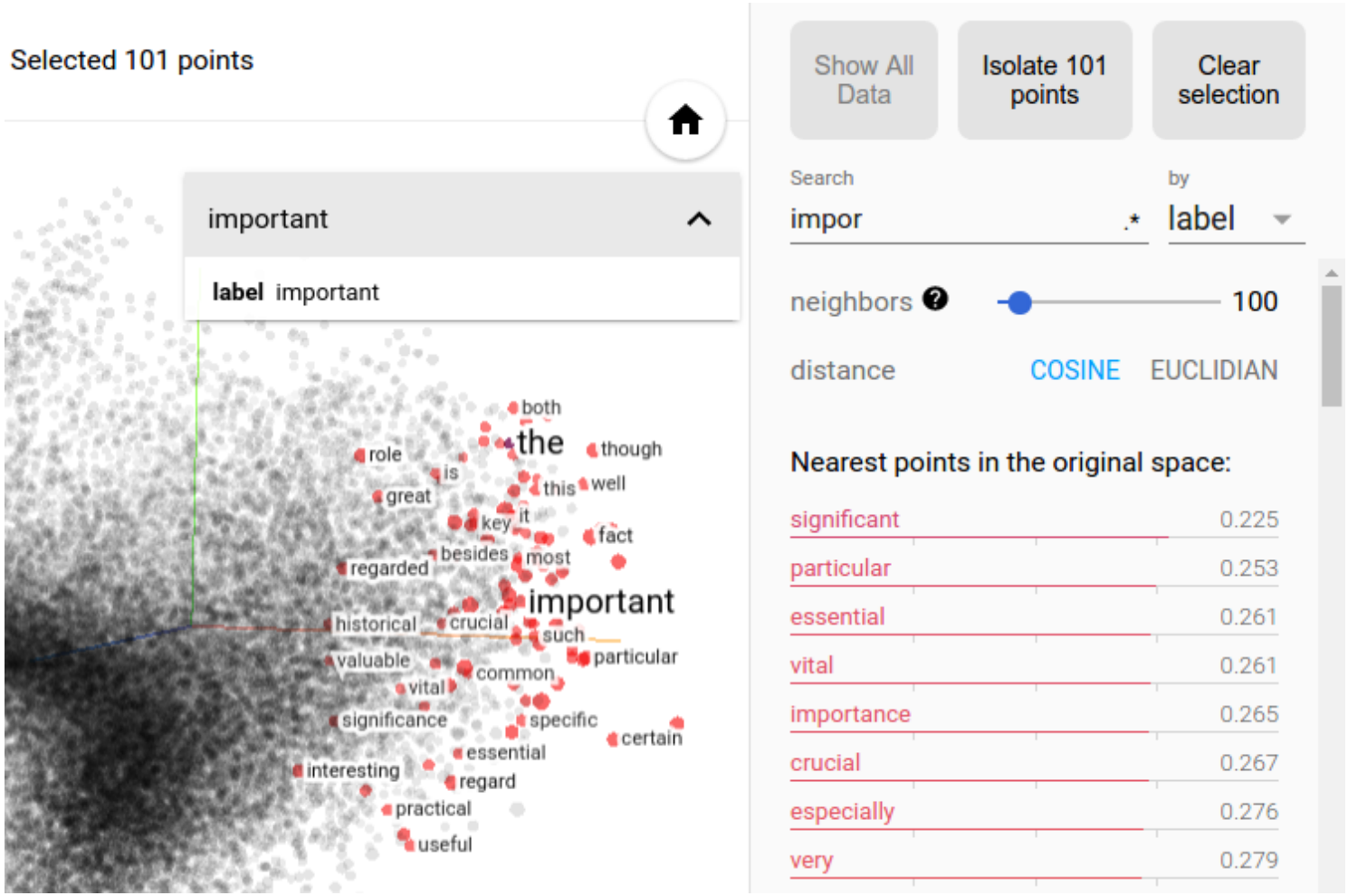
To explore a data set, you can navigate the views in either a 2D or a 3D mode, zooming, rotating, and panning using natural click-and-drag gestures. Clicking on a point causes the right pane to show an explicit textual list of nearest neighbors, along with distances to the current point. The nearest-neighbor points themselves are highlighted on the projection.

Zooming into the cluster gives some information, but it is sometimes more helpful to restrict the view to a subset of points and

perform projections only on those points. To do so, you can select points in multiple ways:

1. After clicking on a point, its nearest neighbors are also selected.
2. After a search, the points matching the query are selected.
3. Enabling selection, clicking on a point and dragging defines a selection sphere.

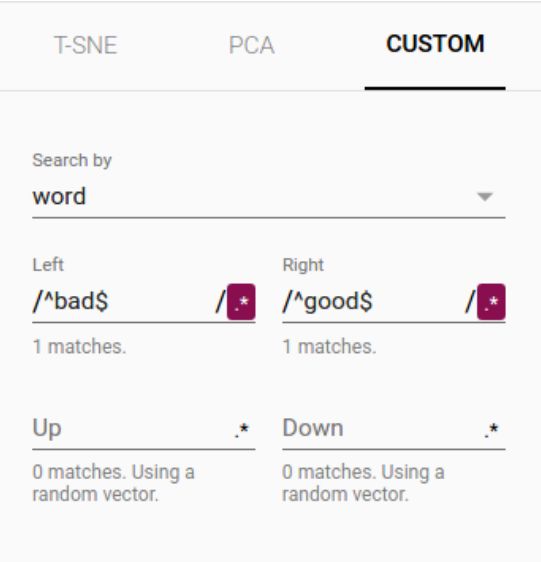
After selecting a set of points, you can isolate those points for further analysis on their own with the "Isolate Points" button in the Inspector pane on the right hand side.



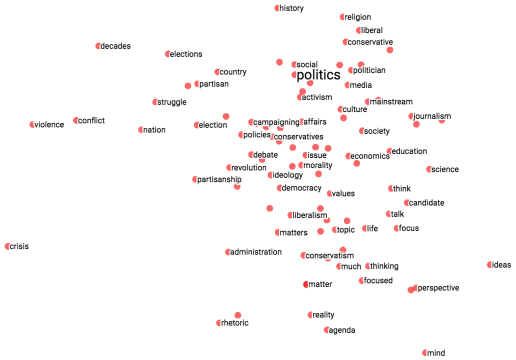
Selection of the nearest neighbors of "important" in a word embedding dataset.

The combination of filtering with custom projection can be powerful. Below, we filtered the 100 nearest neighbors of "politics" and projected them onto the "best" - "worst" vector as an x axis. The y axis is random.

You can see that on the right side we have "ideas", "science", "perspective", "journalism" while on the left we have "crisis", "violence" and "conflict".



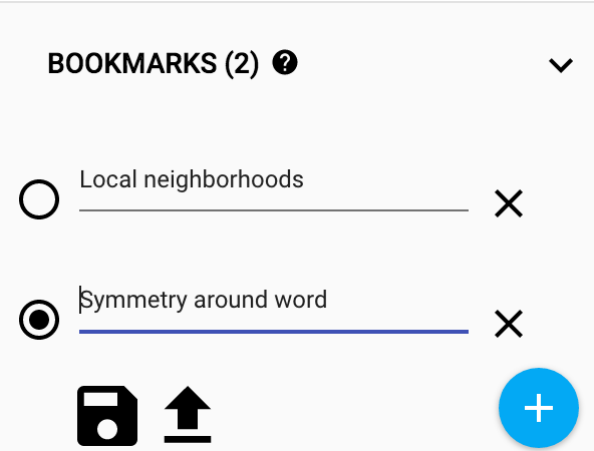
Custom projection controls.



Custom projection of neighbors of "politics" onto "best" - "worst" vector.

Collaborative Features

To share your findings, you can use the bookmark panel in the bottom right corner and save the current state (including computed coordinates of any projection) as a small file. The Projector can then be pointed to a set of one or more of these files, producing the panel below. Other users can then walk through a sequence of bookmarks.



Last updated March 8, 2017.