"UBC-ALM: Combining k-NN with SVD for WSD"

Michael Brutz

CU-Boulder Department of Applied Mathematics

21 March 2013
This 2007 paper by Agirre and Lacalle looks at representing senses of a word with feature vectors and combining two mathematical methods to do WSD using those vectors.
Outline

1. Feature Vectors
2. SVD
3. k-Nearest Neighbors
4. Paper’s Results
The idea is to represent a word with a vector (a list of numbers).
The idea is to represent a word with a vector (a list of numbers).

One of the simplest ways to do this is what is known as the bag-of-words approach.
Feature Vectors

If you have the sentence:

"The cat ran."

Then the bag-of-words representation for the words in this sentence is to have a vector where the only non-zero entries are:

the = 1, cat = 1, ran = 1

basically, just having a value of 1 for every position in the vector corresponding to a word that appears in the sentence.
Feature Vectors

To make this into a vector, all we do is associate each entry in the vector with a word.

For the "The cat ran." example, this vector would look like

$$A_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$$

Again, where the 1's are in the slots for "the", "cat", and "ran", and all the others are 0.
Feature Vectors

When we collect a set of such vectors into a single object like so

\[
\begin{pmatrix}
A_1 \\
A_2
\end{pmatrix} =
\begin{pmatrix}
1 & \vdots \\
0 & 1 \\
\vdots & 0 \\
1 & \vdots \\
0 & 1 \\
\vdots & \vdots \\
1 & 0
\end{pmatrix}
\]
Feature Vectors

We have what is called a matrix.

\[
\begin{pmatrix}
A_1 & A_2 \\
\end{pmatrix} = \begin{pmatrix}
1 & \vdots & 1 \\
0 & 1 & \vdots & 0 \\
1 & \vdots & 1 & \vdots \\
1 & 0 & 1 & \vdots \\
\end{pmatrix}
\]
Feature Vectors

I’m only using two vectors for illustration, you can include any number and still have a matrix.

\[
\begin{pmatrix}
A_1 & A_2 & \ldots & A_n
\end{pmatrix} = \\
\begin{pmatrix}
1 & : & : \\
0 & 1 & 0 \\
: & 0 & 1 \\
1 & : & \ldots & : \\
0 & 1 & : & \ldots \\
1 & 0 & 1
\end{pmatrix}
\]
This leads us into SVD.
SVD stands for Singular Value Decomposition.
SVD stands for Singular Value Decomposition.

In mathy terms, it’s where you take a matrix, $A$, and break it up into the product of three special matrices.

$A = U \Sigma V^T$

$U$ contains the left singular vectors of the matrix $A$, $\Sigma$ contains its singular values, and $V$ contains its right singular vectors.
This is what it looks like for a 3 by 3 matrix.

\[
\begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{pmatrix}
= 
\begin{pmatrix}
    u_1 & u_2 & u_3
\end{pmatrix}
\begin{pmatrix}
    \sigma_1 & 0 & 0 \\
    0 & \sigma_2 & 0 \\
    0 & 0 & \sigma_3
\end{pmatrix}
\begin{pmatrix}
    - & v_1 & - \\
    - & v_2 & - \\
    - & v_3 & -
\end{pmatrix}
\]
The terms in the middle matrix, the $\sigma$’s, are what are called the "singular values".

\[
\begin{pmatrix}
  a_{11} & a_{12} & a_{13} \\
  a_{21} & a_{22} & a_{23} \\
  a_{31} & a_{32} & a_{33}
\end{pmatrix} = \\
\begin{pmatrix}
  \sigma_1 & 0 & 0 \\
  0 & \sigma_2 & 0 \\
  0 & 0 & \sigma_3
\end{pmatrix}
\begin{pmatrix}
  u_1 \\
  u_2 \\
  u_3
\end{pmatrix}
\begin{pmatrix}
  - & v_1 & - \\
  - & v_2 & - \\
  - & v_3 & -
\end{pmatrix}
\]
When we delete the singular values the farthest along the diagonal, we still have a good approximation to our original matrix $A$

$$
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\approx
\begin{pmatrix}
\sigma_1 & 0 & 0 \\
0 & \sigma_2 & 0 \\
0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
- & v_1 & - \\
- & v_2 & - \\
- & v_3 & -
\end{pmatrix}
$$
This implies we can approximate the three column vectors of the $A$ matrix by only using the first two $u$ vectors.

$$
\begin{pmatrix}
    a_{11} & a_{12} & a_{13} \\
    a_{21} & a_{22} & a_{23} \\
    a_{31} & a_{32} & a_{33}
\end{pmatrix}
\approx
\begin{pmatrix}
    \sigma_1 & 0 & 0 \\
    0 & \sigma_2 & 0 \\
    0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
    - & v_1 & - \\
    - & v_2 & - \\
    - & v_3 & -
\end{pmatrix}
$$
This implies we can approximate the three column vectors of the $A$ matrix by only using the first two $u$ vectors.

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} \\
a_{21} & a_{22} & a_{23} \\
a_{31} & a_{32} & a_{33}
\end{pmatrix}
\approx
\begin{pmatrix}
| | \\
u_1 \\
u_2 \\
u_3
| |
\end{pmatrix}
\begin{pmatrix}
\sigma_1 & 0 & 0 \\
0 & \sigma_2 & 0 \\
0 & 0 & 0
\end{pmatrix}
\begin{pmatrix}
| | \\
- v_1 \\
- v_2 \\
- v_3
| |
\end{pmatrix}
\]

You have to know how matrix multiplication works to see this, but the idea of using SVD to represent more vectors with fewer is the important thing.
So what do these approximations do?
Let’s ask Big Bird!
Grayscale images are represented with numbers that determine by how bright to make each pixel, meaning this 104 by 138 pixel image is actually represented by a matrix.
Using 1 Singular Value
Using 5 Singular Values
Using 40 Singular Values
Using All 104 Singular Values
As you can see, we can get a pretty good approximation to what all the vectors comprising the image are doing, using only a fraction of the singular vectors.
Moreover, with regards to WSD it isn’t that we want this

Using 40 Singular Values
We actually want this

Using 5 Singular Values
The reason is that while using fewer singular values blurs the image, this is tantamount to adding in unseen word counts with respect to our vector description of words.

Using 5 Singular Values
It’s a double whammy! Not only do we have fewer vectors to deal with, but we also get guesstimates to word counts we haven’t seen in collecting data, but suspect are relevant.

Using 5 Singular Values
So instead of using simple bag-of-words vectors, we instead use these singular vectors to describe senses of a word as they help with sparsity of data issues.
The last piece of the puzzle is to use $k$-NN ($k$-Nearest Neighbors) to disambiguate words.
Before understanding how it works, one needs to realize that vectors have a geometric interpretation as points in space.
For example, take the vector

\[(0.4 \quad 0.6)\]
For example, take the vector

\[
\begin{pmatrix}
0.4 \\
0.6
\end{pmatrix}
\]

This can be represented graphically as
Graphically Representing (0.4, 0.6)
For the sake of illustration, say that this "X" represents the word "bass" and we are trying to find the sense for it.
Now add in the points that have been tagged in training our WSD classifier.
Let $\propto$ represent the "fish" version and the eighth-notes represent the music version for feature vectors from a hand-tagged corpus.
The way k-NN works is to assign the label to the target word that is shared the most by its k nearest hand-tagged neighbors.
For example, if $k = 3$ then in this example we would say that the target word has the fish sense.
If $k = 5$, then the music sense.
In the paper, all of these approaches had some extra bells and whistles thrown on them (e.g. the feature vectors weren’t just bag-of-words, and they used a weighted k-NN), but nothing really worth getting into.

If you’ve understood what I’ve said so far, then you understand what they were fundamentally doing in the paper.
Paper’s Results

They compared their WSD classifier to the others entered in the SemEval-2007 on the Lexical Sample and All-Words tasks, and overall did pretty well:

<table>
<thead>
<tr>
<th>Task</th>
<th>Method</th>
<th>Rank</th>
<th>rec.</th>
<th>prec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LS</td>
<td>Best</td>
<td>1</td>
<td>0.887</td>
<td>0.887</td>
</tr>
<tr>
<td>LS</td>
<td>UBC-ALM</td>
<td>2</td>
<td>0.869</td>
<td>0.869</td>
</tr>
<tr>
<td>LS</td>
<td>Baseline</td>
<td>-</td>
<td>0.780</td>
<td>0.780</td>
</tr>
<tr>
<td>AW</td>
<td>Best</td>
<td>1</td>
<td>0.591</td>
<td>0.591</td>
</tr>
<tr>
<td>AW</td>
<td>k-NN combination</td>
<td>5</td>
<td>0.544</td>
<td>0.544</td>
</tr>
<tr>
<td>AW</td>
<td>Baseline</td>
<td>-</td>
<td>0.514</td>
<td>0.514</td>
</tr>
</tbody>
</table>
Questions?

Thank you for your time.