Abstract

Research at the intersection of language and other modalities, most notably vision, is becoming increasingly important in natural language processing. We introduce a toolkit that can be used to obtain feature representations for visual and auditory information. MMF-EAT is an easy-to-use Python toolkit, which has been developed with the purpose of making non-linguistic modalities more accessible to natural language processing researchers.

1 Introduction

Distributional models are built on the assumption that the meaning of a word is represented as a distribution over others (Turney and Pantel, 2010; Clark, 2015), which implies that they suffer from the grounding problem (Harnad, 1990). That is, they do not account for the fact that human semantic knowledge is grounded in the perceptual system (Louwerse, 2008). There has been a lot of interest within the Natural Language Processing community for making use of extra-linguistic perceptual information, much of it in a subfield called multi-modal semantics. Such multi-modal models outperform language-only models on a range of tasks, including modelling semantic similarity and relatedness (Bruni et al., 2014; Silberer and Lapata, 2014), improving lexical entailment (Kiela et al., 2015b), predicting compositionality (Roller and Schulte im Walde, 2013), bilingual lexicon induction (Bergsma and Van Durme, 2011) and metaphor identification (Shutova et al., 2016). Although most of this work has relied on vision for the perceptual input, recent approaches have also used auditory (Lopopolo and van Miltenburg, 2015; Kiela and Clark, 2015) and even olfactory (Kiela et al., 2015a) information.

In this demonstration paper, we describe MMF-EAT, a Python toolkit that makes it easy to obtain images and sound files and extract visual or auditory features from them. The toolkit includes two standalone command-line tools that do not require any knowledge of the Python programming language: one that can be used for automatically obtaining files from a variety of sources, including Google, Bing and FreeSound (miner.py); and one that can be used for extracting different types of features from directories of data files (extract.py). In addition, the package comes with code for manipulating multi-modal spaces and several demos to illustrate the wide range of applications. The toolkit is open source under the BSD license and available at https://github.com/douwekiela/mmfeat.

2 Background

2.1 Bag of multi-modal words

Although it is possible to ground distributional semantics in perception using e.g. co-occurrence patterns of image tags (Baroni and Lenci, 2008) or surrogates of human semantic knowledge such as feature norms (Andrews et al., 2009), the de facto method for grounding representations in perception has relied on processing raw image data (Baroni, 2016). The traditional method for obtaining visual representations (Feng and Lapata, 2010; Leong and Mihalcea, 2011; Bruni et al., 2011) has been to apply the bag-of-visual-words (BoVW) approach (Sivic and Zisserman, 2003). The method can be described as follows:

1. obtain relevant images for a word or set of words;
2. for each image, get local feature descriptors;
3. cluster feature descriptors with k-means to find the centroids, a.k.a. the “visual words”;
4. quantize the local descriptors by comparing them to the cluster centroids; and
5. combine relevant image representations into an overall visual representation for a word.

The local feature descriptors in step (2) tend to be variants of the dense scale-invariant feature transform (SIFT) algorithm (Lowe, 2004), where an image is laid out as a dense grid and feature descriptors are computed for each keypoint.

A similar method has recently been applied to the auditory modality (Lopopolo and van Miltenburg, 2015; Kiela and Clark, 2015), using sound files from FreeSound (Font et al., 2013). Bag-of-audio-words (BoAW) uses mel-frequency cepstral coefficients (MFCCs) (O’Shaughnessy, 1987) for the local descriptors, although other local frame representations may also be used. In MFCC, frequency bands are spaced along the mel scale (Stevens et al., 1937), which has the advantage that it approximates human auditory perception more closely than e.g. linearly-spaced frequency bands.

2.2 Convolutional neural networks

In computer vision, the BoVW method has been superseded by deep convolutional neural networks (CNNs) (LeCun et al., 1998; Krizhevsky et al., 2012). Kiela and Bottou (2014) showed that such networks learn high-quality representations that can successfully be transferred to natural language processing tasks. Their method works as follows:

1. obtain relevant images for a word or set of words;
2. for each image, do a forward pass through a CNN trained on an image recognition task and extract the pre-softmax layer;
3. combine relevant image representations into an overall visual representation for a word.

They used the pre-softmax layer (referred to as FC7) from a CNN trained by Oquab et al. (2014), which was an adaptation of the well-known CNN by Krizhevsky et al. (2012) that played a key role in the deep learning revolution in computer vision (Razavian et al., 2014; LeCun et al., 2015). Such CNN-derived representations perform much better than BoVW features and have since been used in a variety of NLP applications (Kiela et al., 2015c; Lazaridou et al., 2015; Shutova et al., 2016; Bulat et al., 2016).

2.3 Related work

The process for obtaining perceptual representations thus involves three distinct steps: obtaining files relevant to words or phrases, obtaining representations for the files, and aggregating these into visual or auditory representations. To our knowledge, this is the first toolkit that spans this entire process. There are libraries that cover some of these steps. Notably, VSEM (Bruni et al., 2013) is a Matlab library for visual semantics representation that implements BoVW and useful functionality for manipulating visual representations. DISSECT (Dinu et al., 2013) is a toolkit for distributional compositional semantics that makes it easy to work with (textual) distributional spaces. Lopopolo and van Miltenburg (2015) have also released their code for obtaining BoAW representations.1

3 MMFeat Overview

The MMFeat toolkit is written in Python. There are two command-line tools (described below) for obtaining files and extracting representations that do not require any knowledge of Python. The Python interface maintains a modular structure and contains the following modules:

- mmfeat.miner
- mmfeat.bow
- mmfeat.cnn
- mmfeat.space

Source files (images or sounds) can be obtained with the miner module, although this is not a requirement: it is straightforward to build an index of a data directory that matches words or phrases with relevant files. The miner module automatically generates this index, a Python dictionary mapping labels to lists of filenames, which is stored as a Python pickle file index.pkl in the data directory. The index is used by the bow and cnn modules, which together form the core of the package for obtaining perceptual representations. The space package allows for the manipulation and combination of multi-modal spaces.

miner Three data sources are currently supported: Google Images2 (GoogleMiner), Bing Images3 (BingMiner) and FreeSound4 (FreeSoundMiner). All three of them require API keys, 1https://github.com/evanmiltenburg/soundmodels-iwcs
2https://images.google.com
3https://www.bing.com/images
4https://www.freesound.org
which can be obtained online and are stored in the
miner.yaml settings file in the root folder.

**bow**  The bag-of-words methods are contained in
this module. BoVW and BoAW are accessible
through the mmfeat.bow.vw and mmfeat.bow.aw
modules respectively, through the BoVW and
BoAW classes. These classes obtain feature de-
scriptors and perform clustering and quantization
through a standard set of methods. BoVW uses
dense SIFT for its local feature descriptors; BoAW
uses MFCC. The modules also contain an inter-
face for loading local feature descriptors from
Matlab, allowing for simple integraton with e.g.
VLFeat[^5]. The centroids obtained by the clustering
(sometimes also called the “codebook”) are stored
in the data directory for re-use at a later stage.

**cnn**  The CNN module uses Python bindings
to the Caffe deep learning framework (Jia et
al., 2014). It supports the pre-trained reference
adaptation of AlexNet (Krizhevsky et al., 2012),
GoogLeNet (Szegedy et al., 2015) and VGGNet
(Simonyan and Zisserman, 2015). The interface is
identical to the **bow** interface.

**space**  An additional module is provided for
making it easy to manipulate perceptual represen-
tations. The module contains methods for aggre-
gating image or sound file representations into vi-
sual or auditory representations; combining per-
ceptual representations with textual representa-
tions into multi-modal ones; computing nearest
neighbors and similarity scores; and calculating
Spearman $\rho_s$ correlation scores relative to human
similarity and relatedness judgments.

### 3.1 Dependencies

MMFeat has the following dependencies: *scipy*,
*scikit-learn* and *numpy*. These are standard Python
libraries that are easy to install using your favorite
package manager. The BoAW module additionally
requires *librosa[^6]* to obtain MFCC descriptors.
The CNN module requires Caffe[^7]. It is recom-
mended to make use of Caffe’s GPU support, if
available, for increased processing speeds. More
detailed installation instructions are provided in the
readme file online and in the documentation
of the respective projects.

[^5]: http://www.vlfeat.org
[^6]: https://github.com/bmcfee/librosa
[^7]: http://caffe.berkeleyvision.org

### 4 Tools

MMFeat comes with two easy-to-use command-
line tools for those unfamiliar with the Python pro-
gramming language.

#### 4.1 Mining: miner.py

The **miner.py** tool takes three arguments: the data
source (bing, google or freesound), a query file
that contains a line-by-line list of queries, and a
data directory to store the mined image or sound
files in. Its usage is as follows:

```bash
miner.py {bing,google,freesound} \ 
query_file data_dir [-n int]
```

The -n option can be used to specify the number of
images to download per query. The following ex-
amples show how to use the tool to get 10 images
from Bing and 100 sound files from FreeSound for
the queries “dog” and “cat”:

```bash
$ echo -e "dog\ncat" > queries.txt
$ python miner.py -n 10 bing \ 
queries.txt ./img_data_dir
$ python miner.py -n 100 freesound \ 
queries.txt ./sound_data_dir
```

#### 4.2 Feature extraction: extract.py

The **extract.py** tool takes three arguments: the type
of model to apply (boaw, bovw or cnn), the data
directory where relevant files and the index are
stored, and the output file where the representa-
tions are written to. Its usage is as follows:

```bash
extract.py [-k int] [-c string] \ 
[-o {pickle,json,csv}] [-s float] \ 
[-m {vgg,alexnet,googlenet}] \ 
{boaw,bovw,cnn} data_dir out_file
```

The -k option sets the number of clusters to use in
the bag of words methods (the $k$ in k-means). The
-c option allows for pointing to an existing code-
book, if available. The -s option allows for sub-
sampling the number of files to use for the cluster-
ing process (which can require significant amounts
of memory) and is in the range 0-1. The tool can
output representation in Python pickle, JSON and
CSV formats. The following examples show how
the three models can easily be applied:

```bash
python extract.py -k 100 -s 0.1 bovw \ 
./img_data_dir ./output_vectors.pkl
python extract.py -gpu -o json cnn \ 
./img_data_dir ./output_vectors.json
python extract.py -k 300 -s 0.5 -o csv \ 
boaw ./sound_data_dir ./out_vecs.csv
```
5 Getting Started

The command-line tools mirror the Python interface, which allows for more fine-grained control over the process. In what follows, we walk through an example illustrating the process. The code should be self-explanatory.

Mining The first step is to mine some images from Google Images:

```python
datadir = '/path/to/data'
words = ['dog', 'cat']
n_images = 10
from mmfeat.miner import *
m = GoogleMiner(datadir, 
    '/path/to/miner.yaml')
m.miner.getResults(words, n_images)
m.miner.save()
```

Applying models We then apply both the BoVW and CNN models, in a manner familiar to scikit-learn users, by calling the fit() method:

```python
from mmfeat.bow import *
from mmfeat.cnn import *
b = BoVW(k=100, subsample=0.1)
c = CNN(modelType='alexnet', gpu=True)
b.load(data_dir)
b.fit()
c.load(data_dir)
c.fit()
```

Building the space We subsequently construct the aggregated space of visual representations and print these to the screen:

```python
from mmfeat.space import *
for lkp in [b.toLookup(), c.toLookup()]:
    vs = AggSpace(lkp, 'mean')
    print vs.space
```

These short examples are meant to show how one can straightforwardly obtain perceptual representations that can be applied in a wide variety of experiments.

6 Demos

To illustrate the range of possible applications, the toolkit comes with a set of demonstrations of its usage. The following demos are available:

1-Similarity and relatedness The demo downloads images for the concepts in the well-known MEN (Bruni et al., 2012) and SimLex-999 (Hill et al., 2014) datasets, obtains CNN-derived visual representations and calculates the Spearman \( \rho_s \) correlations for textual, visual and multi-modal representations.

2-ESP game To illustrate that it is not necessary to mine images or sound files and that an existing data directory can be used, this demo builds an index for the ESP Game dataset (Von Ahn and Dabbish, 2004) and obtains and stores CNN representations for future use in other applications.

3-Matlab interface To show that local feature descriptors from Matlab can be used, this demo contains Matlab code (`run_dsift.m`) that uses VLFeat to obtain descriptors, which are then used in the BoVW model to obtain visual representations.

4- Instrument clustering The demo downloads sound files from FreeSound for a set of instruments and applies BoAW. The mean auditory representations are clustered and the cluster assignments are reported to the screen, showing similar instruments in similar clusters.

5-Image dispersion This demo obtains images for the concepts of elephant and happiness and applies BoVW. It then shows that the former has a lower image dispersion score and is consequently more concrete than the latter, as described in Kiela et al. (2014).

7 Conclusions

The field of natural language processing has broadened in scope to address increasingly challenging tasks. While the core NLP tasks will remain predominantly focused on linguistic input, it is important to address the fact that humans acquire and apply language in perceptually rich environments. Moving towards human-level AI will require the integration and modeling of multiple modalities beyond language.

Advances in multi-modal semantics show how textual information can fruitfully be combined with other modalities, opening up many avenues for further exploration. Some NLP researchers may consider non-textual modalities challenging or outside of their area of expertise. We hope that this toolkit enables them in carrying out research that uses extra-linguistic input.

Acknowledgments

The author was supported by EPSRC grant EP/I037512/1 and would like to thank Anita Verő, Stephen Clark and the reviewers for helpful suggestions.
References


