NLP Raport: Opening word embeddings with dictionary learning

Maksymilian Szmelczyński[1]

[1] *University of Warsaw, Warsaw, Poland*

June 30, 2020

1 Introduction

This project's main goal was to visualize in an insightful way GloVe word embeddings, which are dense, continuous vector representations of words obtained using an unsupervised learning algorithm and co-occurrence statistics from a large corpus of text. There are few ways to accomplish the task of visualization of the word embeddings:

- Nearest neighbors
- t-SNE
- subset PCA

All of these options have some drawbacks. The nearest neighbor approach produces only a single scalar number encapsulating closeness information, while it is clear that word embeddings can be related in much more multidimensional and intricate ways. t-SNE reduces word vectors to a very low-dimensional space in a non-linear way. It can reveal some interesting clusters of words but is hard to interpret due to nonlinearity, it squashes all the information to just a few dimensions, most often two, and is not able to expose potential linear structures inherent in the word embeddings that one can see in word analogy tasks, like "work + profession = worker". And lastly, the PCA approach performed on the subset of word pairs, which have certain relations, e.g adjective-superlative. Its main drawback is that one has to manually chose proper word pairs that stay in specific relation to each other that we want to explore.

The linear substructures generated using the subset PCA approach and semantic content of arithmetic operations that they expose provide strong motivation to automatically discover the factors which these underlying word vectors are composed of.

As a way to remedy some of the limitations and drawbacks of methods listed above in this project the method called Sparse Dictionary Learning was applied.

Sparse coding is a representation learning method that aims at finding a sparse representation of the input data in the form of a linear combination of basic elements and also simultanously founding the form of these basic elements themselves. The elements are called atoms and together they compose a dictionary.

The main goal was to see if word vectors can be decomposed as a linear combination of a sparse subset of interpretable word factors and if so, open these word embeddings to gain some insight into their inherent, not easily visible (because of dense, continuous nature of their representation), properties and visualize obtained results in interesting, compressed and informing fashion. If these goals can be achieved with dictionary learning, then it would be potentially a powerful visualization tool for understanding word embeddings.

2 Implementation

The solution was implemented in Python using sklearn's dictionary learning module. The first step was to find a dictionary, a set of atoms, that can best be used to represent data using sparse code. The optimization problem that was solved had the form:

Table 1: *Vector-factor manipulations. The word vectors' average norm is 6.9. The learned word factors all have unit norm. Note that $2nd$ not $1st$ nearest neigbors are displayed.*

Manipulations	2nd Nearest Neighbor
$V_{swim} + 5F_{profession}$	$V_{swimmer}$
$V_{mine} + 5F_{profession}$	V_{miner}
$V_{school} + 5F_{profession}$	$V_{teacher}$
$V_{sport} + 5F_{profession}$	$V_{athlete}$

$$\begin{aligned} \underset{U,V}{\arg\min} \quad & 0.5 \cdot \|Y - UV\|_2^2 + alpha \cdot \|U\|_1 \\ \text{s.t.} \quad & \|V_k\|_2 = 1 \\ & 0 \le k < ncomponents \end{aligned} \tag{1}$$

Different values for parameters controlling the size of the dictionary (the number of atoms/word factors) and alpha (controls the sparsity of obtained code, the higher its value the sparser the code) were tried, but ultimately there were set to 1000 and 0.5 respectively. 1000 word factors and alpha set to 0.5 seemed to be a combination of parameters that showed a good tradeoff between reconstruction error and interpretability of obtained word factors.

3 Results

After learning a dictionary, a subset of learned word factors (atoms) was manually labeled based on top-activation words associated with a particular factor. Then one can decompose word vectors into a linear combination of named words factors (Figure 1) and subsequently explore in a more transparent way structure inherent in this word embeddings performing some manipulations and finally visualizing obtained results in various ways.

In Figure 2 there is a table containing a few chosen word factors along with top-activation words. We can see that words are grouped into interpretable, semantically, or syntactically related clusters.

By adding particular word factors into GloVe word embeddings we can perform interesting manipulations, transforming vectors in a predictable and desirable way (Figure 1).

Figure 5 and Figure 4 show different levels of activation of specified word factors in pairs of words that are syntactically related in the same way. We can see that word factor activations reflect this difference.

4 Conclusions

Results obtained in this project show that sparse dictionary learning can extract elementary factors from dense, continuous word vectors. By using the learned factors, we can meaningfully decompose or open, otherwise hard to interpret, word vectors and visualize them in many interesting ways. It can guide our intuition and help in better understanding their inner workings and also maybe give some hints in cases when they don't work as expected, e.g. in word analogy tasks.

It is interesting to see that word vectors can be thought of, at least to some extent, as a linear combination of word factors that contain often clear semantic or syntactic meaning and that this linear structure emerges from unsupervised learning based on co-occurrence statistics.

This project can be expanded in several ways and directions. One would be performing some form of clustering on learned word factors because it seems that always a few different word factors share similar meanings and could be group together. This could allow for even better, more robust, and complete visualizations. A natural extension of this project would be to apply the same technique on polish word embeddings to see how it would perform in this setting, having in mind that it is much more morphologically rich language. Another interesting direction would be to try to apply sparse dictionary learning to contextual word embeddings like ones produced by BERT or GPT-2 models.

In summary, sparse dictionary learning is an interesting visualization technique that can be used to open up dense, continuous word embeddings holding potential in developing our understanding of, otherwise hard to interpret, word representations.

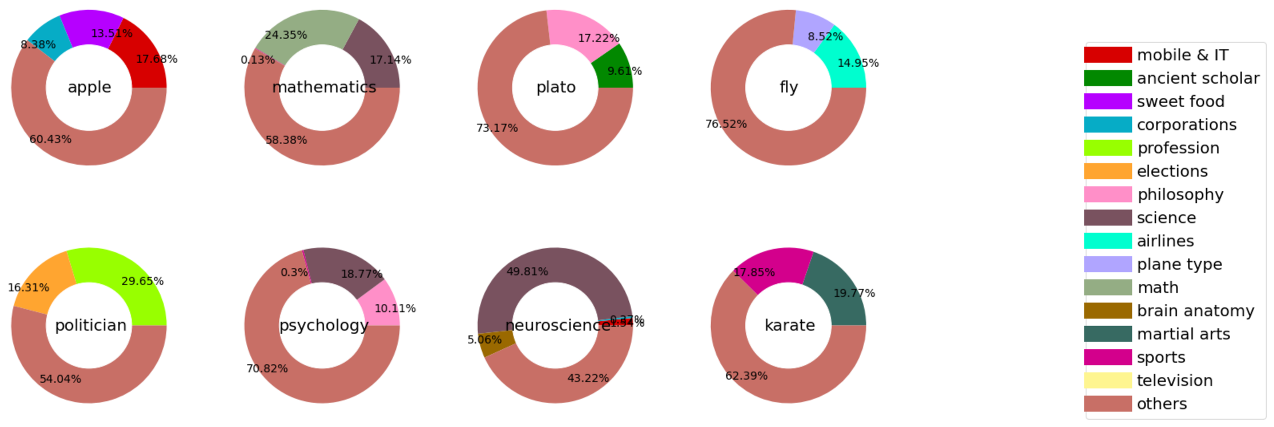

Figure 1: *Word vectors can be decomposed into a sparse linear combination of word factors.*

Factors	**Top activation words**
0 *"space"*	*astronauts, astronaut, soyuz, shuttle, iss, spacewalks, spacewalk, endeavour cosmonauts, atlantis, space, mir, shenzhou, cosmonaut, nasa, shenzhou-7 spacecraft, shenzhou-6, liftoff, undocked*
974 *"elections"*	*seats, constituency, re-elected, by-election, governorships, first-past-the-post, elections, councillors elected, constituencies, seat, by-elections, election, independents, parliamentary, seanad unopposed, votes, reelected, legislative*
152 *"food"*	*fried, salads, pork, chicken, grilled, soups, steak, sausage sandwiches, meats, dishes, beef, cooked, fries, meat, roast salad, stew, sausages, tacos*
66 *"philosophy"*	*marxism, positivism, anarchism, epistemology, aristotelian, kantian, philosophers, hegelian existentialism, hegel, empiricism, philosophy, materialist, rationalism, philosopher, metaphysics philosophical, kant, structuralism, nietzsche*
667 *"insects"*	*insects, beetles, ants, insect, larvae, spiders, caterpillars, moths termites, mosquitoes, prey, mites, bees, aphids, worms, larval invertebrates, butterflies, pests, snakes*
394 *"science"*	*neuroscience, immunology, peer-reviewed, biology, microbiology, epidemiology, neurology, genetics pharmacology, parasitology, psychiatry, endocrinology, physiology, oncology, biophysics, informatics biochemistry, virology, sciences, entomology*
193 *"computer"*	*powerpc, x86, pentium, risc, cpus, microprocessors, mips, microprocessor celeron, xeon, processors, processor, 68000, athlon, x86-64, cpu 64-bit, z80, 8-bit, mainframes*
129 *"injuries"*	*wounding, wounded, injuring, killed, injured, blast, grenade, assailants explosion, injures, gunbattle, gunfire, gunmen, ambush, roadside, detonated exploded, attacker, kills, bomber*

Figure 2: *Set of learned factors with its top-activation words. Factors were named manually.*

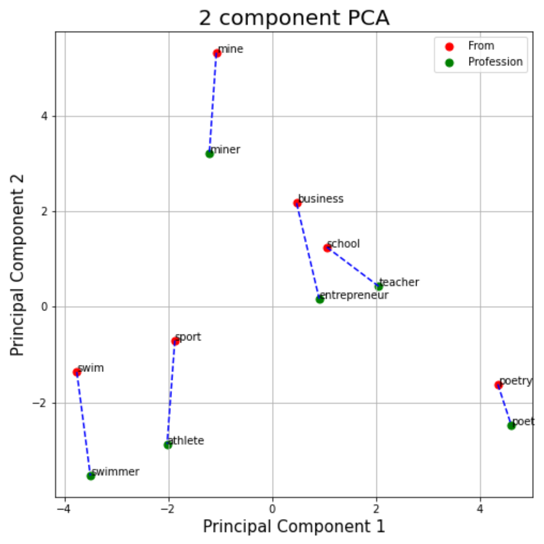

Figure 3: *PCA visualization of a word analogy task: "profession", which can be automatically generated by the "profession" word factor.*

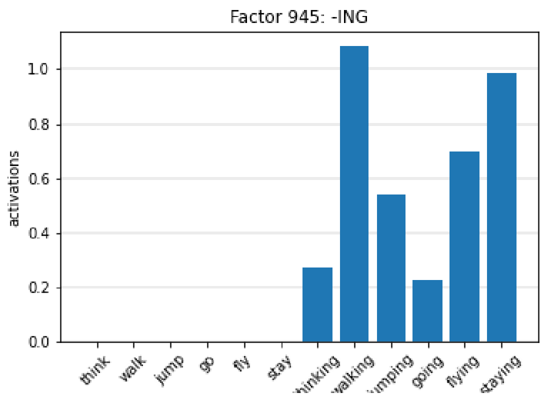

Figure 4: *"-ing" factor's activation w.r.t. a selected set of words contain verbs in basic form and its present continous tense*

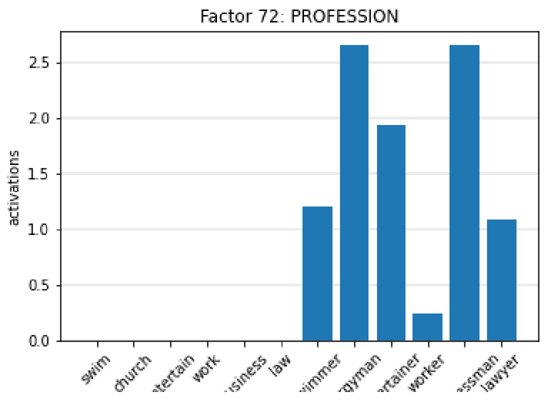

Figure 5: *"profession" factor's activation w.r.t. a selected set of words contain action-related words and their profession form.*