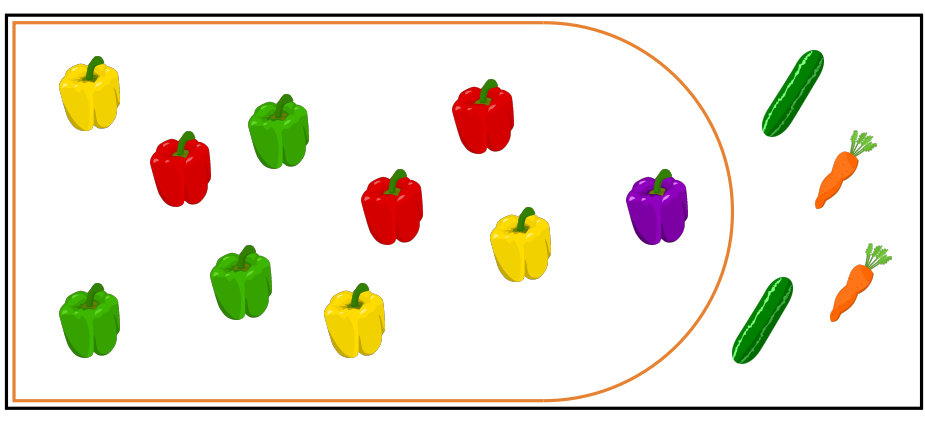
Autoencoding Pixies

Guy Emerson

The aim of distributional semantics is to learn the meanings of words from a corpus of text. Below are six instances of the word “pepper”, from the English Wikipedia. From these, a human might infer that peppers are plants cultivated for food. Distributional semantics develops algorithms to learn this kind of information automatically.

... annuum includes the “bell	pepper”	variety, which is sold in ...
... Ideal growing conditions for	peppers	include a sunny position with ...
... capsaicin in a pungent (hot)	pepper	is concentrated in blisters ...
... Pickled or marinated	peppers	are frequently added to ...
... In Hungary, sweet yellow	peppers	– along with tomatoes – are ...
... genome. Over 75% of the	pepper	genome is composed of ...

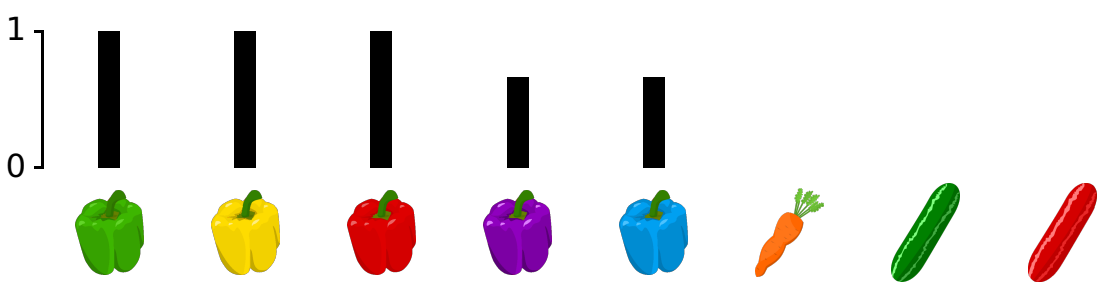
The aim of formal semantics is to develop mathematical models of meaning. A widely used approach is *truth-conditional semantics*, which describes meaning in terms of *truth*, relative to a *model*. A model represents the world – for example, the toy model below consists of fourteen entities. The meaning of the word “pepper” is the set of entities for which it is true, indicated by the orange line.



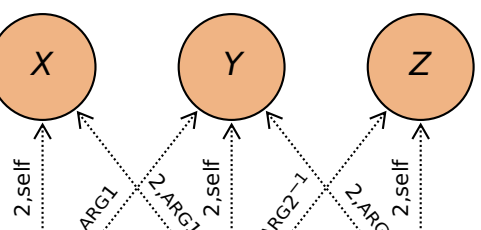
In this poster, I will show:

- How to learn truth-conditional semantics from a corpus
- How this can be done efficiently
- How this improves performance on semantic tasks

Representing the meaning of a word as a set of entities is awkward if we don’t already know all entities. We can instead view the meaning of a word as a way of *classifying* each entity, based on its features. This is shown below, where each image is not an individual entity, but rather a representation of an entity’s features. I will call such a representation a *pixie*. The meaning of a word is a *truth-conditional function*, mapping pixies to probabilities of truth.



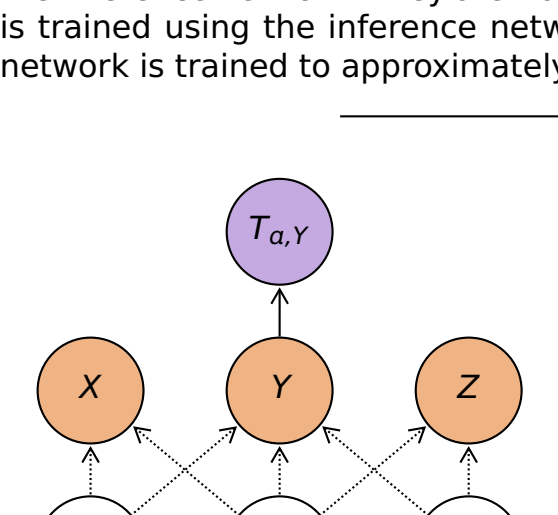
The aim is to define a machine learning model that uses truth-conditional functions, and train the model on a corpus.



Given a corpus parsed into semantic dependency graphs, as shown here, the model can learn from precise linguistic relationships. A suitable corpus is WikiWoods, a parsed version of the English Wikipedia.

So, more precisely, the aim is to define a model that can generate dependency graphs, and optimise the model so that it generates graphs which closely match the graphs observed in the corpus.

Below is a *probabilistic graphical model* that generates dependency graphs. The idea is that each observed dependency graph describes an unobserved situation consisting of multiple entities. For example, the graph above describes a situation with a chef, a pepper, and a cutting event. Each of these latent entities is represented by a pixie.



Each node is a random variable. Nodes in the top row are entities, each represented by a pixie in the space $\mathcal{X}$. Nodes in the middle row are truth values, either true ($\top$) or false ($\bot$). The rectangular *plate* denotes repetition of nodes: for each word r in the vocabulary $\mathcal{V}$, it can be true or false for each of the entities. Finally, nodes in the bottom row are words. For each entity, the model chooses one word out of all the words that are true.

This model has two parts that must be optimised: the *world model*, which captures how pixies co-occur with each other; and the *lexical model*, which consists of a truth-conditional function for each word.

A probabilistic generative model can be trained by maximising the likelihood of the observed data. This requires inferring the latent pixies, but unfortunately, exact calculations are intractable. I propose using *amortised variational inference* to approximately infer the latent pixies.

Further, I propose using a *graph-convolutional network* to perform variational inference. The input is a dependency graph, and the output is a predicted pixie for each node. At the bottom, the network is initialised with an embedding for each word. (For example, p, q, r could be “chef”, “cut”, “pepper”.) These pass through two convolutional layers, to get the predicted pixies at the top. In each layer, each node is updated based on its neighbours. This allows the predictions to be contextualised: for example, peppers are not cut in the same way that grass is cut.

The *Pixie Autoencoder* is the combination of the generative model and the inference network. They are trained together: the generative model is trained using the inference network’s predictions and the inference network is trained to approximately fit the generative model.

The Pixie Autoencoder can be used to perform logical inference. Given a dependency graph, we could ask what other words would be true of the same entities. For example, if a chef is cutting a pepper, is that cutting event also a slicing event? Or a mowing event? (Presumably yes and no, respectively.) To answer such questions, the inference network can be applied to the dependency graph, and then a truth-conditional function from the generative model can be applied to the relevant pixie node. This gives the truth value node at the top, where a could be “slice” or “mow”.

The GS2011 dataset evaluates semantic similarity in context, comparing pair of verbs with the same subject and object. For example, a chef cutting a pepper is similar to a chef slicing a pepper, but not very similar to a chef mowing a pepper. The dataset contains 199 pairs, with similarity judgements from multiple human annotators. Given below are Spearman rank correlations with the average judgements.

Model	Correlation
Word2Vec	.348
BERT	.446
Pixie Autoencoder	.504

The Pixie Autoencoder is compared against two established models, Word2Vec (which represents each word as a vector), and BERT (which predicts contextualised vectors).

BERT has been found effective on a range of other tasks. It is a larger model than the Pixie Autoencoder, and has been trained on more data. The higher performance of the Pixie Autoencoder suggests that linguistic structure is helpful for fine-grained semantic tasks.

For full details, and for further results, read the paper:
`https://arxiv.org/abs/2005.02991`