Gavagai

Published: Fri 22 April 2016
By Federico Vaggi
In Machine Learning.
tags: video games machine learning dota2

Last time I talked about Dota2, I covered the evolution of the metagame across patches. This time, I'll focus on analyzing the way pros draft - but looking at it using tools from NLP in a very novel way.

Introduction:

How would you teach someone what the word 'rabbit' means? You could go out in a field, and every time a rabbit shows up you could point to it and exclaim 'rabbit'. However, even this highly simplified vignette assumes that you are trying to explain the concept of rabbit to someone who has a mental model of the world very similar to yours. Now, imagine that instead of teaching the meaning of words to a human who sees the world as you do, you are trying to teach them to a computer. Where do you start?

This is where Word Vectors come in. The key assumption behind word vectors is that words which occur in similar contexts have similar meanings - and this is known as the Distributional Hypothesis.

In the case of linguistics, this means that if you find the word 'negotiate' in similar contexts as the word 'bargain', their meaning is probably similar. Using word vectors, it means that if you calculate the cosine similarity (a measure of how similar two vectors are) between the word vector for 'negotiate' and 'bargain' you'll get a value close to 1. Another neat property of word vectors is that they allow us to reason by analogy: a common example is that (King - Man) ~= (Queen - Woman). For an awesome example of the power of word vectors, check out an analysis of every single reddit post done using spacy.

From Word Vectors to Hero Vectors:

How can word vectors help us understand Dota2 teams? Imagine that you treat every Dota2 team as a sentence, with heroes making up the words. For example, this would be a valid sentence in Dota2ese:

Batrider Disruptor Leshrac Naga Siren Weaver

and so would this:

Dragon Knight Weaver Crystal Maiden Windranger Treant Protector

Recall the Distributional Hypothesis that we discussed above. Instead of trying to understand what words mean by looking at the sentences in which they occur, we want to understand what heroes are like by looking at the kinds of team they show up in. Concretely, if we see that heroes like **Witch Doctor** and **Lion** tend to show up in similar teams, this indicates that they have similar roles.

We will use datdota as our database of drafts, and the excellent gensim library to learn our hero vectors.

When learning our hero vectors, we have to specify how many dimensions we want to use to represent our heroes. In general, more dimensions means that we get higher quality representations, but we require more computing power and more data.

For example, after training on our dataset, this is what the hero vector for **Shadowfiend** looks like.

```
[-0.06813218, -0.00902375, 0.10162564, -0.01908037, 0.03013835,
0.16538762, 0.03104097, 0.02496031, -0.16785616, 0.3313826 ,
-0.21904311, -0.07945664, 0.19140202, 0.12729862, 0.36308175,
0.19962946, 0.13561839, 0.23637122, -0.32607114, 0.05647549,
0.09655968, -0.21899879, 0.04926173, 0.12474103, 0.14504923,
0.06281823, 0.14728694, -0.03583163, -0.00227163, 0.1205247 ,
0.01127683, 0.01522848, 0.13806115, 0.0216765 , 0.13671157,
-0.1683237 , 0.00408782, 0.10514087, -0.17610508, 0.04697264,
-0.03406512, -0.14956233, 0.20201634, 0.00907436, -0.05804597,
-0.00481437, 0.11493918, -0.07718568, -0.13443205, -0.01155808]
```

The hero **Shadow Fiend** is now represented as a point in 50 dimensional space, but that's still not that useful: we turned a word we can understand into 50 numbers that don't seem to mean anything. However, I promise that we can do some really cool things: for example - let's look at the other heroes whose vectors are most similar to that **Shadow Fiend**:

1. **Queen of Pain**: 0.9340388774871826

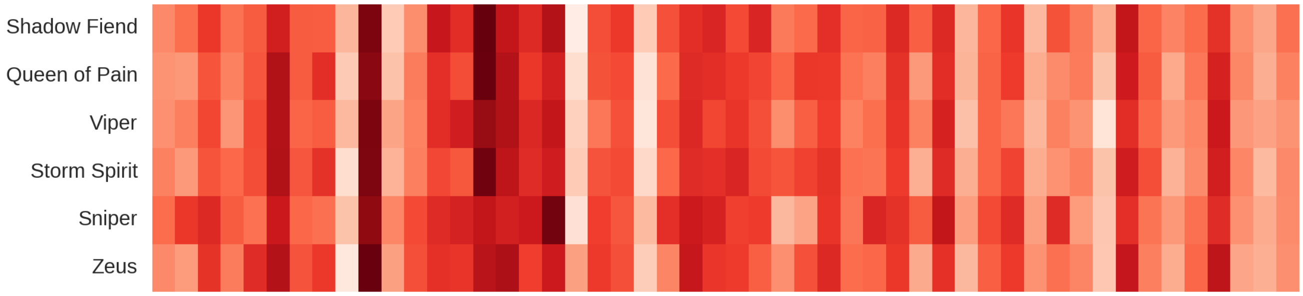
2. **Storm Spirit**: 0.9170020818710327

3. **Viper**: 0.9082884788513184

4. **Sniper**: 0.8958033919334412

5. **Zeus**: 0.8526902794837952

And here it is in a more visual form - each row corresponds to the vector for a particular hero:



Wow - those actually all make a lot of sense. Let's try a tougher task now and see if we can use hero vectors to reason about analogies. **Lion** is to **Anti Mage** as **Witch Doctor** is to ....:

1. **Spectre**: 0.9638912677764893

2. **Phantom Lancer**: 0.9185065031051636

3. **Phantom Assassin**: 0.9039324522018433

4. **Morphling**: 0.858444333076477

5. **Lifestealer**: 0.8570600748062134

Again, pretty neat! While these synergies won't exactly let you outdraft PPD, hero vectors can pick up on the notion of pairing a ranged support with a melee carry.

A Global View:

It's possible to plot the hero vectors of every single hero at the same time, but the results can be a bit overwhelming.

In that figure, every column corresponds to a hero vector, and we have sorted rows and columns in a dendogram. Unless you are used to looking at those kinds of graphs all the time, it's not very obvious which heroes form groups and which heroes are similar to each other.

Ignoring the hero vectors for a minute, we can cut the dendogram (the tree-like structure at the top of the previous graph) and look at which heroes end up close to each other.

We see that we have recovered some pretty meaningful (although not flawless) groups. We can notice an 'offlane-like' group with Doom, Nyx, Dark Seer, Bristleback and Centaur. We have a hard carry group with Drow Ranger, Slark, Phantom Lancer and Lifestealer. We have a 'strength support' group with Wraith King, Tusk and Abbadon.

There's one more way to visualize our groups. We can project our 50 dimensional dataset in 2d using t-SNE](https://lvdmaaten.github.io/tsne/). t-SNE is a very clever technique where we look for a lower dimensional representation that still maintains as much of the high level structure as possible.

After manipulating the data and plotting it in 2d, here is what we get:

The hero colors come from the groups we identified above. Again - notice that there are easily discrete clusters that are quite meaningful.

What do the vectors mean?:

What exactly do the numbers that we obtain for each hero mean though? For example, what does it mean that that the first dimension of **Shadow Fiend** has value -0.06813218? Usually, we'd just say that it is a 'latent feature' and leave it at that, but in this case, we can do a little bit more.

The dota2 gamepedia assigns a series of roles for each hero. They are of course debatable, and heroes can often change from support to carry across different patch - but it's still a good starting point. Here is a simple table of all the roles assigned to each hero.

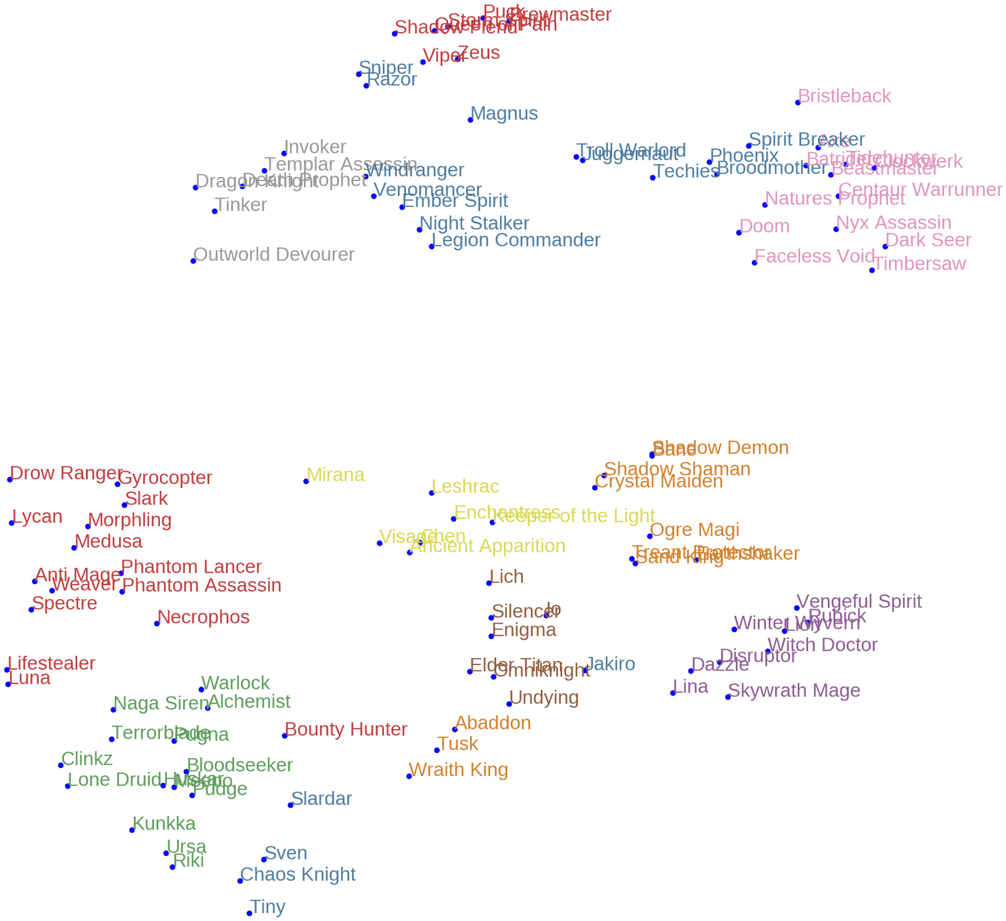
Now, we have all the ingredients for the next step. Recall that each hero is described by 50 numbers (from our embedding procedure) and that hero has certain roles (from the dota2 gamepedia).

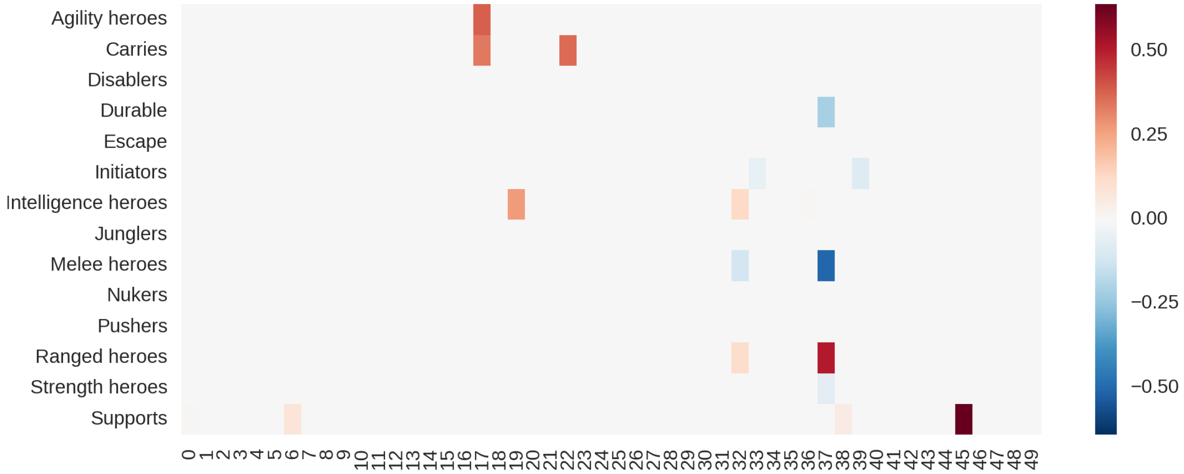
What we want to do now is to see if there is any relationship between the embedding and a hero's role. There's lots of ways to do this - but perhaps the easiest way is to use logistic regression.
For the math nerds - to insure that we get a sparse set of basis, we will also use a strong L1 penalty (which, as a bonus, regularizes our predictions).

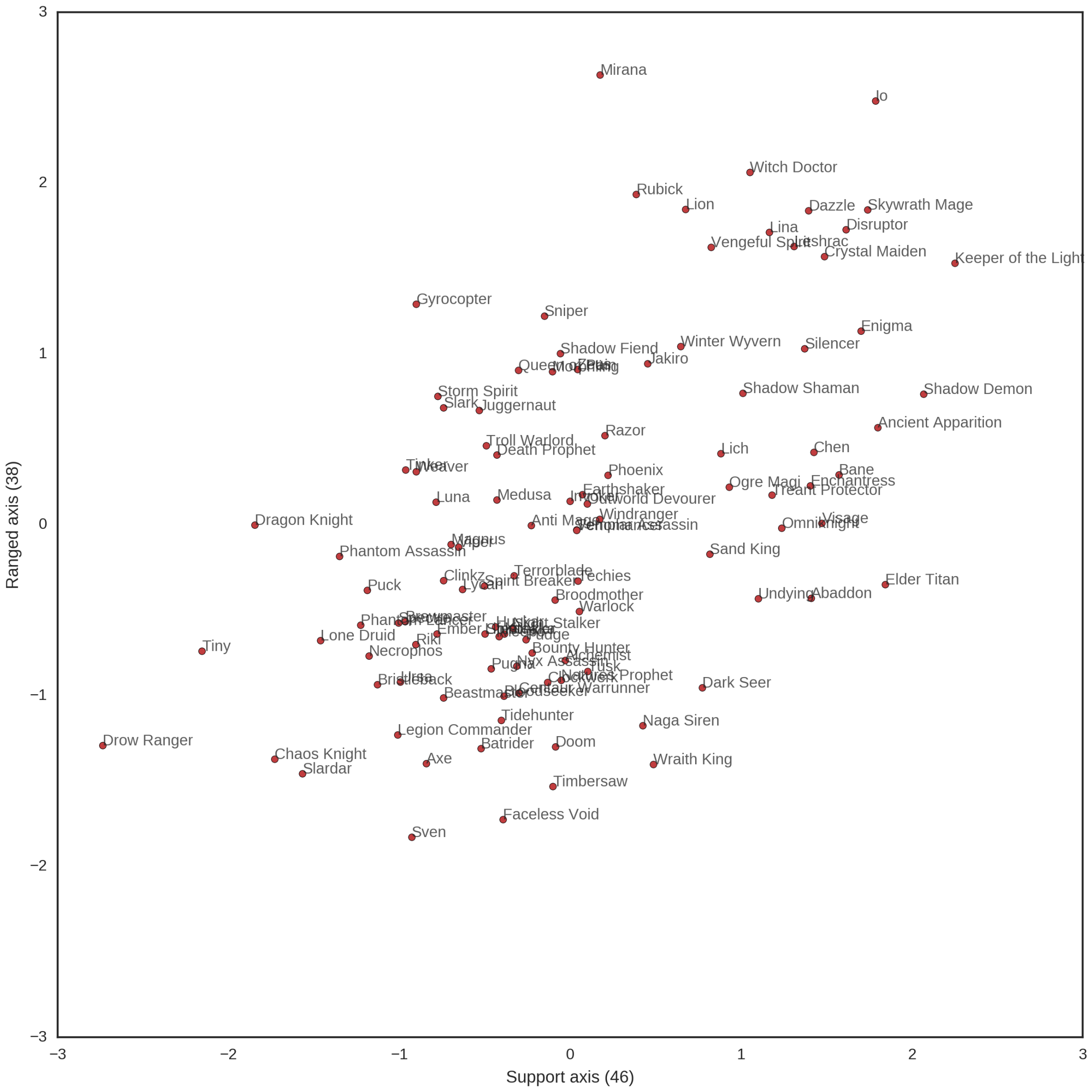
Here are the results of that:

What does that mean, in practice? We see that the 46th vector is strongly associated with being a support, and the 38th vector is strongly associated with being a ranged hero. Let's plot the 38th and 46th vector of all heroes:

As we can see - heroes that have higher values on axis 46, are a lot more likely to be support heroes. Similarly, heroes that have higher values on axis 38 are more likely to be ranged - you may notice that the two axis are somewhat correlated (but not perfectly so) - as support heroes are a lot more likely to be ranged. Incidentally, this correctly identifies **Undying**, **Elder Titan** and **Abbadon** as melee supports (high support, low ranged) but incorrectly identifies **Drow Ranger** (and, to a certain extent, **Clinkz** as melee heroes.


Puck
Brewmaster
Storm Spirit
Queen of Pain
Shadow Fiend
Viper
Zeus
Sniper
Razor
Magnus
Bristleback
Invoker
Templar Assassin
Windranger
Dragon Knight
Death Prophet
Venomancer
Ember Spirit
Tinker
Night Stalker
Legion Commander
Outworld Devourer
Troll Warlord
Juggernaut
Phoenix
Techies
Broodmother
Spirit Breaker
Axe
Tidehunter
Batrider
Clockwerk
Beastmaster
Natures Prophet
Centaur Warrunner
Doom
Nyx Assassin
Dark Seer
Faceless Void
Timbersaw
Shadow Demon
Bane
Shadow Shaman
Crystal Maiden
Drow Ranger
Gyrocopter
Mirana
Leshrac
Slark
Lycan
Morphling
Medusa
Enchantress
Keeper of the Light
Visage
Chen
Ancient Apparition
Ogre Magi
Treant Protector
Earthshaker
Sand King
Phantom Lancer
Anti Mage
Weaver
Phantom Assassin
Spectre
Necrophos
Lich
Silencer
Io
Enigma
Vengeful Spirit
Rubick
Winter Wyvern
Lion
Witch Doctor
Disruptor
Dazzle
Lina
Skywrath Mage
Lifestealer
Luna
Elder Titan
Omniknight
Jakiro
Undying
Warlock
Naga Siren
Alchemist
Terrorblade
Pugna
Bounty Hunter
Abaddon
Tusk
Clinkz
Bloodseeker
Lone Druid
Huskar
Meepo
Pudge
Wraith King
Slardar
Kunkka
Ursa
Riki
Sven
Chaos Knight
Tiny


Agility heroes
Carries
Disablers
Durable
Escape
Initiators
Intelligence heroes
Junglers
Melee heroes
Nukers
Pushers
Ranged heroes
Strength heroes
Supports
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
0.50
0.25
0.00
−0.25
−0.50



Conclusion:

When training those embeddings, we did not use any information about Dota2: as far as the computer is concerned, each draft is an arbitrary sequence of symbols without any meaning... yet, starting from that, we were able to recover informative representations which have meaningful high level interpretation.
We treated Dota2 drafts as a foreign language that we do not speak, and, by looking at patterns in how those words are used, managed to figure out what the words mean relative to each other.

Further, by connecting the features that we learned in an unsupervised way to class roles, we were even able to give an interpretation to our latent features.

Proudly powered by Pelican, which takes great advantage of Python.

The theme is by Smashing Magazine, thanks!