Stitching Together Vector Spaces

Linas Vepštas

Draft of 17 June 2018 - First Draft

Abstract

Applying machine learning to linguistics to extract both syntax and semantics, ideally via unsupervised algorithms is the goal of an incresingly popular quest. At this time, vector-space based approaches seem to be the best suited for this, and are thus increasingly commonplace and quite popular. This includes systems such as word2vec and and GloVe as exemplars. These systems fall short, however, as they fail to expose the syntactic structure of natural language. The reason for this, and how to move beyond this state of affairs, is discussed in this text.

The key observation made here is that there is more than just one vector space onto which words can be mapped. The different vector spaces are necessaarily related to one-another through syntactic relations. When a set of words are clustered into a grammatical category, this clustering must be made in a consistent fashion, for each of the vector spaces. The vector spaces must be "stitched together" in a consistent fashion; this consistency condition is exactly the syntax of the grammar. The stitching obeys a certain set of axioms, which are the sheaf axioms. The best possible stitching together can be found by minimizing an overall cost function.

Introduction

Not written. See next section.

Also, the last half of this text is not written.

Meaning

So here's one approach to meaning. It is already clear that disjuncts are correlated with meaning, so one provisional approach might be to assign each disjunct a unique meaning. Alternately, this can be used as a doorway to the intentional meaning of a word.

Consider the phrases "the big balloon", "the red balloon", "the small ballon"... The pseudo-disjuncts on balloon in these three cases would be "the- big-" "the - red-" and "the- small-" (plus an additional connector to the verb). Examining this connector-by-connector, we expect that the MI for the word pair (the, balloon) to be small, while the MI for the word-pairs (big, balloon), (red, balloon) and (small, balloon) to be large(r). Its thus tempting to identify the set {big, red, small} as the set of intentional attributes associated with “balloon”. The strength of the MI values to each of the connectors might be taken as a judgement of how much that attribute is prototypical of the object (see other section on “prototype theory”).

The disjuncts associated with “balloon” will also connect to a verb. These verb connectors may be taken as another set of intentional attributes, for example {floats, drifts, rose, popped}. It should be possible to distinguish these as an orthogonal set of attributes, in that one might observe “the- red- floats+” and “the- red- drifts+” but never observe “floats- drifts+”.

Meaning bibliography:

- “The Molecular Level of Lexical Semantics”, EA Nida, (1997) International Journal of Lexicography, 10(4): 265–274. https://www.academia.edu/36534355/The_Molecular_Level_of_Lexica

Vector Algebra

Here’s the deal ... vectors.

Word vectors

The last section introduces the idea of decomposable basis vectors. Let try to make this more precise (a cleaned-up version of this and above belongs in the (unfinished) sheaf paper.). First, review the notion of a vector, and then show how to make vectors from word-disjuncts, N-grams, and skip-grams. Then review the idea of tensoring.

So, the textbook standard definition of a vector $\vec{v}$ is

$$\vec{v} = a_1\widehat{e}_1 + a_2\widehat{e}_2 + \cdots + a_n\widehat{e}_n$$

The a_k are conventionally taken to be numbers; here, real numbers. The $\widehat{e}_k$ are called “basis vectors”. They have the property of having unit length; that is $\|\widehat{e}_k\| = 1$. For every word w, one can create several kinds of vectors. A widely-known vector is the “N-gram”, where each $\widehat{e}_k$ corresponds to the N-word context in which the word w was observed, and a_k is the count of the number of observations. Thus, for example, given the toy corpus: "*John knew the bird was there. John heard the bird. John saw the bird. Susan saw the bird too. Mary saw it also*". Setting $w = bird$ and N=3, and the requirement that the word be in the middle of the N-gram, one has

$$\begin{aligned}\widehat{e}_1 &= [the * was]\\ \widehat{e}_2 &= [the * .]\\ \widehat{e}_3 &= [the * too]\end{aligned}$$

and observation counts $a_1 = 1$, $a_2 = 2$, $a_3 = 1$. Here, $a_2 = 2$ because there are two sentences containing the word-sequence “the bird.” In more compact form, one may write

$$\overrightarrow{bird} = 1\,[the * was] + 2\,[the * .] + 1\,[the * too]$$

This can be extended in an obvious way to a larger corpus, to a larger N and one can drop the requirement that the word be in the middle. Skip-grams are similar, but allow some words to be skipped, and have a complex algorithm to determine when and how words can be skipped.

Disjuncts behave in a very similar manner. The word-pair counting and the MST pipeline creates disjuncts and the associated counts. For the above corpus, the disjuncts might possibly resemble these:

$$\widehat{e}_1 = the- \ \& \ was+$$
$$\widehat{e}_2 = the- \ \& \ .+$$
$$\widehat{e}_3 = the- \ \& \ too+$$

These disjuncts are essentially identical to the N-gram example above; only the notation is different. One might hope that the disjuncts coming out from the MST pipeline might actually be of higher quality. The support for this hope is several decades of research results on MST parsing of natural language. The point here is that N-grams, skip-grams and word-disjunct vectors are quite similar to one another, and that, perhaps, word-disjunct vectors are just a tiny bit more linguistically accurate.

What does one do with vectors? Well, one can apply any vector-based algorithm you can wrap your mind around. Some classics are clustering; SVM or K-means. Here, one commonly starts with a vector dot-product, leading to a cosine-distance metric. Given two vectors $\vec{a} = a_1\widehat{e}_1 + a_2\widehat{e}_2 + \cdots + a_n\widehat{e}_n$ and $\vec{b} = b_1\widehat{e}_1 + b_2\widehat{e}_2 + \cdots + b_n\widehat{e}_n$, the dot-product is given by

$$\vec{a} \cdot \vec{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$$

The cosine angle between these is given by

$$\|\vec{a}\| \left\|\vec{b}\right\| \cos\theta = \vec{a} \cdot \vec{b}$$

where, as usual, the l_2 Hilbert-space norm is used: $\|\vec{a}\| = \sqrt{\vec{a} \cdot \vec{a}}$. The closer that θ is to zero, the more parallel the two vectors are. There are plenty of other interesting ways of measuring the similarity of vectors. The point here is that if the corpus included the sentences "*John heard the crow. John saw the crow.*" and one obtained a vector for $\overrightarrow{crow}$, one might typically discover that $\overrightarrow{crow}$ is similar to $\overrightarrow{bird}$: the cosine angle gives a concrete technology for measuring similarity.

Suppose one has a large number of vectors for a large number of words. Similarity metrics that assign a number to the informal idea of similarity then opens the door to automatically classifying words into bins, according to their similarity. A common favorite is the K-means clustering algorithm, which can take a large number of vectors, and assign them to one of K different bins, with all words in the same bin being quite 'similar'. One can then make general hand-waving arguments that these clusters correspond to 'grammatical classes'. The actual data, the actual results make this obvious: a quick skim of the clusters makes it clear that some clusters are dominated by nous, and others by verbs.

There are other approaches, too. Popular ones are the various deep learning approaches applied to neural nets. Word2Vec is a particularly famous one; its now a tensorflow tutorial. The algorithmic details are interesting, and sophisticated. However, conceptually, its not really all that different from what is described above: one obtains a collection of vectors, and then one pumps the vectors through a neural-net classifier, to obtain word-classes.

The entire point of this section is to drive home the idea that, whatever one can do with N-grams or with skip-grams, one can also do with word-disjunct vectors. At the appropriate conceptual level, they are really very, very similar to one-another.

As a minor side-point, one can hold out some hope that perhaps word-disjunct vectors are of (slightly) higher quality than skip-grams. It seems like they could be or should be, as reinforced by decades of MST-linguistics results. If one takes some stock, off-the-shelf linguistics-K-means algorithm, or some stock, off-the-shelf Word2vec algorithm, and instead plugs in word-disjunct vectors wherever the skip-grams or N-grams might go, then what happens? Maybe, possibly, one might get results that are an itsy-bitsy teensy-weensy bit better. Maybe. This is not known: there are no published results comparing these. It is reasonable (to me) to expect that the results would be quite similar, with perhaps word-disjunct vectors being just a smidgen higher-quality.

Due diligence suggests that such head-to-head comparisons should be performed and published.

A later sections describes how one can do much better than naive clustering, whether Word2vec or K-means. But first, some additional basic issues need to be reviewed.

Merging vectors

In the above example, it seems obvious that "*bird*" and "*crow*" should be merged: they are grammatically similar, given their vectors. So lets merge them into a class of "*THINGS*". How should that class be created, formally? This is a non-trivial question, since additional merges affect the class. Suppose, for example, the corupus contained the sentence "*John saw the book*". Is it plausible to merge "book" into "THINGS"? If so, then how? One way is to compute the cosine similarity $\cos\left(\overrightarrow{book},\overrightarrow{bird}\right)$ and $\cos\left(\overrightarrow{book},\overrightarrow{crow}\right)$ and merge only if both are large enough. The other is to compute $\cos\left(\overrightarrow{book},\overrightarrow{THINGS}\right)$ but this requires the vector $\overrightarrow{THINGS}$.

There are several ways to create this vector. One way is by explicit vector addition

$$\overrightarrow{THINGS}_{sum} = \overrightarrow{bird} + \overrightarrow{crow}$$

This seems to be an adequate way of creating this grammatical class. Recall that

$$\overrightarrow{bird} = 1\,|the-\ \ \&\, was+\rangle + 2\,|the-\ \ \&\ .+\rangle + 1\,|the-\ \ \&\, too+\rangle$$

where we switched to disjunct notation. The vector for $\overrightarrow{crow}$ is

$$\overrightarrow{crow} = 2\,|the-\ \ \&\ .+\rangle$$

Comparing these two, directly like this, suggests that other merge strategies are possible. For example, since $\overrightarrow{crow}$ has neither $|the-\&was+\rangle$ nor $|the-\&too+\rangle$ on it, maybe these don't belong on the merged class: perhaps an intersection of the basis vectors should be used, so that the combined vector is non-zero only on the intersection of the basis elements. Thus, perhaps a better definition would be

$$\overrightarrow{THINGS}_{intersect} = 4\,|the-\ \&\ .+\rangle$$

This is narrow definition is nice, because we have that

$$\overrightarrow{book} = 1\,|the-\ \&\ .+\rangle$$

and so clearly the cosine between $\overrightarrow{THINGS}_{intersect}$ and $\overrightarrow{book}$ is one, and they're mergeable. By contrast, the cosine between $\overrightarrow{THINGS}_{sum}$ and $\overrightarrow{book}$ is much less than one - which also makes sense, because books are not very bird-like.

Another possibility is to combine some fraction $0 \le \alpha \le 1$ of the basis elements that don't overlap. This suggests a grammatical class of

$$\overrightarrow{THINGS}_{\alpha} = \alpha\,|the-\ \&\ was+\rangle + 4\,|the-\ \&\ .+\rangle + \alpha\,|the-\ \&\ too+\rangle$$

so that

$$\overrightarrow{THINGS}_{\alpha=0} = \overrightarrow{THINGS}_{intersect}$$

while

$$\overrightarrow{THINGS}_{\alpha=1} = \overrightarrow{THINGS}_{sum}$$

That is, for $\alpha = 1$, $\overrightarrow{THINGS}_{\alpha}$ is rather bird-like, whereas for $\alpha = 0$, its more generic.

Note that $\overrightarrow{THINGS}_{\alpha}$ is not a linear combination of $\overrightarrow{bird}$ and $\overrightarrow{crow}$. It is non-linear, and, depending on the actual corpus, could be highly non-linear.

The choice of merge strategy alters the contents of the grammatical clusters. This is not a surprise; the goal here is to illustrate that a number of different merge strategies are possible. it is not *a priori* obvious which is the best.

Merge Strategies and Semantics

The different strategies for merging vectors lead in a natural way to the automated discovery of word-meaning. This can be illustrated by example; although a different example from the above is needed.

Consider the word "*saw*". After observing a sufficient amount of text, one will find a vector $\overrightarrow{saw}$ that contains disjuncts for appropriate for usages as a "cutting tool", "the verb cut", and "the past tense of to see". Suppose that there is an existing class $\overrightarrow{TOOLS}$, consisting of nouns, and that the similarity measure judges that $\overrightarrow{saw}$ and $\overrightarrow{TOOLS}$ are similar. Clearly, a merge strategy of expanding an existing cluster by linear addition is incorrect. The expanded class

$$\overrightarrow{TOOLS}_{linear-expand} = \overrightarrow{TOOLS} + \overrightarrow{saw}$$

is incorrect, or at least, not very good, as it now contains disjuncts for "the verb cut" and "the past tense of to see". Two bad things happen: (a) the noun cluster is polluted with verb-vector components, and (b) the vector has not been factorized, and so "*saw*" cannot also be placed into other clusters as well. One looses the ability to distinguish different semantic meanings based on grammatical usage.

We know that, *a priori*, the correct way of thinking about $\overrightarrow{saw}$ is that it has the components

$$\overrightarrow{saw} = \vec{v}_{tool} + \vec{v}_{past-tense} + \vec{v}_{cutting}$$

However, we do not know (*a priori*) what the components $\vec{v}_{tool}$,$\vec{v}_{past-tense}$ and $\vec{v}_{cutting}$ are. A good merge strategy would be able to factorize them out, to discover these automatically.

The best merge strategy is hardly obvious. The three vectors $\vec{v}_{tool}$,$\vec{v}_{past-tense}$ and $\vec{v}_{cutting}$ are not orthogonal (or rather, in general, won't be), so even if there were pre-existing grammatical classes for these three, an orthogonal decomposeion of $\overrightarrow{saw}$ into components is not unique. What's more, in early stages, a class for $\overrightarrow{TOOLS}$ might exist, but not yet one for $\overrightarrow{CUTTING}$; its not even clear that a class of the form $\overrightarrow{CUTTING}$ might even be formable.

There are several possible non-linear merge strategies that are possible. One is given in the next section. However, the point here is not to propose the best-posible merge strategy, but rather to point out that this is where word-sense disambiguation comes from, and that this is how it can be done. Different word-senses are already encoded in how words are used in sentences. Word-senses can be distinguished by paying attention to the disjuncts attached on a word. Picking the correct merge strategy picks out the word-senses. Picking the wrong strategy blurs them all together.

An example non-linear merge strategy

The earlier section illustrated a non-linear merge, using $\overrightarrow{THINGS}_\alpha$ as an example. This section formalizes that example. The result is a maybe-OK merge strategy, but not an obviously great one. Its worth writing down only because its simple.

The idea starts with the computation of the intersection and the union of the support for the vectors to be merged, and then taking a fraction $0 \le \alpha \le 1$ on the union of the support when merging. This merge strategy seems to at least partly overcome the problem of the erasure of word-senses noted above.

Suppose $\overrightarrow{CLASS}$ is some existing word-class, and $\overrightarrow{word}$ is a word that was judged "mergeable" into $\overrightarrow{CLASS}$. Define the support of a vector $\vec{v}$ as

$$\mathscr{I}_{\vec{v}} = \{\widehat{e}_k \text{ such that } \vec{v} = a_1\widehat{e}_1 + \cdots + a_n\widehat{e}_n \text{ and } a_k \neq 0\}$$

The common support for $\overrightarrow{CLASS}$ and $\overrightarrow{word}$ would then be

$$\mathscr{I}_{\overrightarrow{CLASS}} \cap \mathscr{I}_{\overrightarrow{word}}$$

while the union would be

$$\mathscr{I}_{\overrightarrow{CLASS}} \cup \mathscr{I}_{\overrightarrow{word}}$$

with $\cap$ and $\cup$ denoting set-intersection and set-union, respectively. A plausible merge strategy for $\vec{v} = a_1\widehat{e}_1 + \cdots + a_n\widehat{e}_n$ and $\vec{w} = b_1\widehat{e}_1 + \cdots + b_n\widehat{e}_n$ might be

$$\text{merge}\left(\vec{v}, \vec{w}, \alpha\right) = c_1\widehat{e}_1 + \cdots + c_n\widehat{e}_n$$

with

$$c_k = \begin{cases} a_k + b_k & \text{if } \widehat{e}_k \in \mathscr{I}_{\vec{v}} \\ \alpha b_k & \text{if } \widehat{e}_k \notin \mathscr{I}_{\vec{v}} \end{cases}$$

This is designed so that, if $\vec{v} = \overrightarrow{CLASS}$ and $\vec{w} = \overrightarrow{word}$ then those disjuncts in $\overrightarrow{word}$ that are already in $\overrightarrow{CLASS}$ are folded in, in full, while disjuncts from $\overrightarrow{word}$ that are not yet in $\overrightarrow{CLASS}$ are only folded in a little bit, thus expanding the support for $\overrightarrow{CLASS}$, but not putting a lot of weight into the expanded support.

If $\vec{v}$ is a word, and not a class, and one wishes to for a class for the first time, out of two words, one might instead choose

$$c_k = \begin{cases} a_k + b_k & \text{if } \widehat{e}_k \in \mathscr{I}_{\vec{v}} \cap \mathscr{I}_{\vec{w}} \\ \alpha\left(a_k + b_k\right) & \text{otherwise} \end{cases}$$

What should the 'constant' α be? Certainly, a small α is very conservative, not expanding the meaning very much beyond the overlapping set. Perhaps a variable α would be better; for example, taking $\alpha = \left(\cos\left(\vec{v}, \vec{w}\right) - \beta\right) / \left(1 - \beta\right)$ where β is the smallest cosine that allows merging. This makes the merges more conservative, the more different the words are, but is very accepting when the words are similar.

To avoid expanding the support in unwise ways, perhaps it would be best to assemble a set of mergeable words, first, and only then merge them all together, in one shot, rather than merging pair-wise.

The merge strategy here seems OK, but probably not great. A better merge strategy is proposed below. The point here is that merge strategies that do perform word-sense disambiguation (that is, semantic extraction) are in principle possible.

Syntactic Broadening and Generalization

The act of merging together also broadens or generalizes the syntactic structure of the language. It extracts grammatical generalities in addition to semantic particulars. This is again best illustrated by example.

Take up the previous example, of a nature scene with birds, and consider the verbs "*saw*" and "*heard*" – John, Susan and others are seeing and hearing birds and crows. The word-vectors for "*saw*" and "*heard*" will include the disjuncts:

$$\begin{aligned} \widehat{e}_4 &= \left|John- \ \& \ bird+\right\rangle \\ \widehat{e}_5 &= \left|Susan- \ \& \ bird+\right\rangle \\ \widehat{e}_6 &= \left|John- \ \& \ crow+\right\rangle \end{aligned}$$

Working explicitly from this example, one would have the vector

$$\overrightarrow{saw} = 1\left|John- \ \& \ bird+\right\rangle + 1\left|Susan- \ \& \ bird+\right\rangle + 1\left|John- \ \& \ crow+\right\rangle$$

If "*bird*" and "*crow*" are merged into the grammatical class "*THINGS*", then none of the basis vectors $\widehat{e}_4$, $\widehat{e}_5$ or $\widehat{e}_6$ are entirely appropriate any more. They should be replaced by basis vectors made of the combined class; possibly, for example,

$$\overrightarrow{saw} = 1\,|John- \ \& \, THINGS+\rangle + 1\,|Susan- \ \& \, THINGS+\rangle$$

But there's something odd in the above: Susan never saw a crow, in the sample corpus; so the above has certainly enlarged or generalized the class of things that ar see-able, at least with respect to Susan.

That is, given specific examples, the act of deducing grammatical categories enlarges the generative size of the grammar. This is a technical statement, and needs some exmplanation. Formally, a 'grammar' is a collection of rules that define how sentences can be parsed. A 'language' is the set of all possible sentences that the grammar allows. The original corpus is a sampling of the language; deduction of the class of "*THINGS*" expands the language, because, in this expanded grammar, the sentence "*Susan saw the crow*" is now allowed, is part of the language, whereas previously it was not.

Parse Ranking

In the above, the expansion of the grammar from

$$saw : \ Susan- \ \& \, bird+;$$

to

$$saw : \ Susan- \ \& \, THINGS+;$$

was implied to be an all-or-nothing act. In fact, one can apply a likilihood or probability to this expansion, with the intent of keeping undesirable expansion at bay; of minimizing the risk of incorrect expansion. This is done by replacing the lexical set that is the grammatical class with a proabilisitically-weighted set. That is, instead of writing

$$THINGS : \ \{bird, crow\}$$

as a named set, one could introduce a probability or cost:

$$THINGS : \ \{(bird, h_{bird})\,, (crow, h_{crow})\}$$

with $h_{crow} = -\log_2 p\,(crow)$ and likewise for h_{bird}. The logarithm is used to keep the costs additive, instead of multiplicative; this is a detail that tends to simplify things. The value $p\,(crow)$ is a pseudo-probability; there are several plausible ways to define it. One might consider, for example, setting

$$p\,(crow) = \cos\left(\overrightarrow{THINGS}, \overrightarrow{crow}\right)$$

that indicates how well "*crow*" aligned with the class of "*THINGS*".

But this is naive – too naive: the cosine product $\cos\left(\overrightarrow{THINGS}, \overrightarrow{crow}\right)$ was derived from vectors that used the grammatical context of "*crow*" and "*bird*". It did NOT take into account that, in the corpus, Susan had not yet seen any birds. Its as if the vector for $\overrightarrow{crow}$ is incorrect – it should be taking Susan into account, in some way, but its not, because the word "Susan" is too far away in the sentence. For the disjuncts, "*Susan*" is the subject of a verb, "*crow*" is the object, and there is no direct linkage of subject and object. For N-grams, "*Susan*" is simply too far away from the object, unless N is raised to an obscenely large value. Skip-grams do not materially improve on this issue. In all cases, the subject is too far away removed.

Can this be remedied? Yes, it can. The key observation will be that there are other (kinds of) vectors, besides the vectors given above. All of the vectors discussed so far are best considered to be 'naive' vectors. They are 'naive' because they ignore the structure of the basis elements themselves. Acknowledging this structure allows one to move beyond the narrow constraints of linear algebra. Acknowledging this structure leads to the concept of a "sheaf on a graph" as a lingustically appropriate generalization of a vector space; it will replace the vector spaces, and the linear algebra, by a more general concept. This is, roughly, one of collections of vector spaces, sewn together at the edges. The word "sheaf" comes from the idea that the axioms defining how the sewing-together is to be performed are the same axioms as those of "sheaf theory". But first, before we get to that, some more examples need to be developed.

Disjuncts are Tensors

The issue with 'naive' vectors is that they bring the story to a close. In ordinary linear algebra, the basis vectors $\widehat{e}_k$ are indecomposable, structure-less, atomic. Nothing more can be said about them. However, in linguistics, the basis vectors $\widehat{e}_k$ are not structure-less; they are made out of words! And that changes everything.

The concept of word-vectors, such as N-grams, skip-grams or word-disjunct vectors, is just fine, but needs to be recognized as a flawed oversimplification for the structure of language. They're reasonably OK concepts, for a first pass, giving OK results. Clearly, word2vec and it's cousins have made a big impression on the industry. That counts for something and should not be dismissed. But (I believe that) one can do better by not ignoring the structure of the basis vectors. These ideas are developed next.

An earlier section wrote down the basis vector

$$\widehat{e}_4 = |John- \ \& \ crow+\rangle$$

The notation used in writing this was modelled on the usual Link Grammar notation for disjuncts. A tensor-style notation would be to write

$$\widehat{e}_4 = |John-\rangle \otimes |crow+\rangle$$

which makes it perhaps a bit more clear that $\widehat{e}_4$ consists of several parts. When the 'naive' word-vectors were constructed, these parts were ignored. In particular, their contribution to the similarity was ignored. The can be remedied by defining a different kind of vector, the cross-connector vector.

Cross-connector vectors

Returning to the nature-watching example, the corpus of seven sentences produces a dataset of observed disjuncts that includes the following:

$$\begin{aligned} knew\ &: John- \ \& \ bird+; \\ heard\ &: John- \ \& \ bird+; \\ saw\ &: John- \ \& \ bird+; \\ saw\ &: Susan- \ \& \ bird+; \\ heard\ &: John- \ \& \ crow+; \\ saw\ &: John- \ \& \ crow+; \end{aligned}$$

All of the above are observed exactly once, each. This is in addition to the observations

$$\begin{aligned} bird\ &: the- \ \& \ was+; & \\ bird\ &: the- \ \& \ .+; & \text{(seen twice)} \\ bird\ &: the- \ \& \ too+; & \\ crow\ &: the- \ \& \ .+; & \text{(seen twice)} \end{aligned}$$

We take up the question, again, can the words "*bird*" and "*crow*" be merged into a single common grammatical class? Previously, this determination was made, using the 'naive' word-disjunct vectors

$$\overrightarrow{bird}_{naive} = 1\,|the- \ \& \ was+\rangle + 2\,|the- \ \& \ .+\rangle + 1\,|the- \ \& \ too+\rangle$$

and

$$\overrightarrow{crow}_{naive} = 2\,|the- \ \& \ .+\rangle$$

The verb constructions allow a different kind of vector, that "crosses over":

$$\begin{aligned} \overrightarrow{bird}_{obj} =& 1\,|knew : John- \ \& \ *+\rangle + 1\,|heard : John- \ \& \ *+\rangle \\ & + 1\,|saw : John- \ \& \ *+\rangle + 1\,|saw : Susan- \ \& \ *+\rangle \end{aligned}$$

The wild-card * is an explicit place-holder for the word in the disjunct. The corresponding vector for "*crow*" would be

$$\overrightarrow{crow}_{obj} = 1\,|knew : John- \ \& \ *+\rangle + 1\,|heard : John- \ \& \ *+\rangle$$

Both of these are again vectors, but very different in form and shape from before.

The mergability decision is going to be different, as a result. Consider, for example, the cosines. Computing explicitly:

$$\cos\left(\overrightarrow{bird}_{naive}, \overrightarrow{crow}_{naive}\right) = \frac{4}{\sqrt{6\cdot 4}} = \frac{2}{\sqrt{6}} \approx 0.8165$$

while

$$\cos\left(\overrightarrow{bird}_{obj}, \overrightarrow{crow}_{obj}\right) = \frac{2}{\sqrt{4\cdot 2}} = \frac{1}{\sqrt{2}} \approx 0.7071$$

So how similar are crows and birds? What should the similarity measure be now? Should one take the average of these? Do something else? Perhaps one might consider a total vector:

$$\overrightarrow{bird}_{total} = \overrightarrow{bird}_{naive} + \overrightarrow{bird}_{obj}$$

This is appealing, but for one important property: the subspaces $\overrightarrow{word}_{naive}$ and $\overrightarrow{word}_{obj}$ are always orthogonal to one-another, always, for any word. These are really distinct vector spaces; they don't mix. Also, there are more than just these two. Consider the sentence "*The bird flew away.*" This suggests a vector

$$\overrightarrow{bird}_{subj} = 1\,|flew : *- \ \& \ away+\rangle$$

where the wild-card is now in the first position, not the second. Clearly $\overrightarrow{word}_{subj}$ is always orthgonal to $\overrightarrow{word}_{naive}$ and $\overrightarrow{word}_{obj}$, for any word. For any disjunct of length N, there are at least N distinct, orthogonal vector spaces, because the wild-card can occur in any one of N distinct locations in the disjunct. The wild-card can also occur with a + attachment, or a - attachment, so there are at least $2N$ distinct vector spaces. Finally, if the disjunct has k attachments that are -, and $N-k$ that are +, then the $*-$ can occur in any of m locations, while $*+$ can occur in any of $N-k$ locations. Adding up these possibilities, disjuncts of length N span a total of $N(N+1)$ mututally pair-wise orthogonal subspaces. That's a lot of different subspaces to consider.

Despite this, one expects an overall consistency in the grammatical classification of a word: if one decides that birds are like crows, then the unified grammatical class of THINGS that birds and crows belong to must behave properly, when placed in any particular grammatical context. Before, John heard and saw birds; now John can hear and see THINGS, and this needs to hold true for all of the various possible grammatical relations:

$$\begin{aligned} heard &: John- \ \& \ THINGS+; \\ THINGS &: the- \ \& \ was+; \end{aligned}$$

It must necessarily be the same word-class "*THINGS*" in both of these locations. It is grammatically inconsistent to have these being distinct from one-another.

Thus one concludes: (a) there is more than one vector space available, over which similarity comparisons can be made; (b) the decision to merge must be made consistently over all available vector spaces; (c) the merge itself must still be non-linear, in order to differentiate between different word-senses attached to the same word. How this may be accomplished is written up in the next section.

The important constraint here is that of (b) – that the resulting grammatical classes must be consistent, in all of the syntactic roles that they can occur in. The various syntactic vector spaces are not independent of one-another, but stitch together.

Merge decisions, redux

Should one take the average of these?

Replacing Cosines by Entropy?

Cosines are not additive. And that's a big problem when trying to add together contributions from cross-connectors.

Consider instead a different word-similarity measure. As always, let $N(w,d)$ be the count of the number of times the disjunct d was observed on word w. Define the right-product

$$f(w,u)=\sum_d N(w,d)N(u,d)$$

with the sum ranging over all disjuncts shared in common between the two words. This is just the dot-product for the two vectors $\vec{w}$ and $\vec{u}$ – that is, $f(w,u)=\vec{w}\cdot\vec{u}$ and so the cosine similarity of two word-disjunct vectors is just

$$\cos\left(\vec{w},\vec{u}\right)=\frac{f\left(w,u\right)}{\sqrt{f\left(w,w\right)f\left(u,u\right)}}$$

Consider instead a similar quantity

$$p\left(w,u\right)=\frac{f\left(w,u\right)}{f\left(*,*\right)}$$

where $f\left(*,*\right)=\sum_{w,u}f\left(w,u\right)$ is a normalization, a total count. The quantity $p(w,u)$ can be interpreted as a probability: it clearly sums to one. It is symmetric: $p(w,u)=p(u,w)$ and one can thus have traditional marginal probabilities:

$$p\left(w\right)=p\left(w,*\right)=\sum_u p\left(w,u\right)$$

This suggests a natural form for the mutual information between words:

$$MI_d\left(w,u\right)=\log_2\frac{p\left(w,u\right)}{p\left(w\right)p\left(u\right)}$$

The subscript d on MI_d serves to remind that this variant of mutual information is derived from the disjunct product, and not from word-pair observations. Unlike the word-pair observations, this value of MI is symmetric under word-interchange: the word-order does not matter.

Replacing Cosines by Surprisingness?

Can this work?

Conclusion

Not yet written.

The End

References