Applying Cone Learning to Compositionality

Daoud Clarke

May 9, 2013

This document outlines potential strategies for applying the work on learning entailment relations to compositionality.

1 Kernel-based Approaches

In these approaches, we implicitly define a vector space on strings of words using a kernel. We can then use recognising textual entailment datasets to learn an ordering on this space.

Benefits of this approach:

- Simple to implement
- Very general model of composition

Downsides to the approach:

- Finding a kernel that works well given the size of the training set may be non-trivial
- Not very linguistically motivated
- Requires learning an ordering on the space associated with strings. This is a much bigger space than the space associated with words, so is potentially a much harder problem.

2 Grammarless Composition

In this approach, we determine a partial ordering on the space $(V \otimes V) \oplus V$ which relates pairs of words to words, where V is the vector space of words. We do this by examining contexts of all words and pairs of words, and then ensuring that for every pair a, b of words, that $\hat{a} \otimes \hat{b} \leq \widehat{ab}$, where $\hat{a}$ is the context vector associated with a. We would likely use window-based features rather than dependencies for this approach.

We then use this to induce a partial ordering on the Fock space $F(V)$ by requiring that $\mathbf{u} \leq \mathbf{v}$ implies that $\mathbf{w} \otimes \mathbf{u} \leq \mathbf{w} \otimes \mathbf{v}$ and $\mathbf{u} \otimes \mathbf{w} \leq \mathbf{v} \otimes \mathbf{w}$. This gives us an order on all strings.

Benefits:

- Potentially simple to implement
- Learn word ordering on a small(er) space

Downsides:

- Not linguistically motivated
- How do we resolve conflicts in orderings when comparing long strings?

3 Grammar-based Composition

In this approach we assume we always have a parse for the two sentences we are comparing, and we learn orderings from a treebank. We need to learn an ordering for every pair of types that can compose. For example, we need an ordering for $(V \otimes NP) \oplus V \oplus NP$ where V and NP are the spaces for verbs and noun phrases respectively. The tensor product of these spaces is thus the space for verb phrases, and the ordering tells us how verb phrases relate to the orderings of verbs and noun phrases. Given parses for sentences, we can recursively apply these orderings to compare them.

Benefits:

- More linguistically motivated

Downsides:

- Fairly complex to implement
- Perhaps less robust as requires getting the parse correct

4 Adapting Boolean Semantics to MV Algebras

The Boolean semantics of Keenan and Faltz abstracts Montague semantics to allow the conjunction and disjunction of any constituent to be described elegantly, since every constituent is represented as a Boolean algebra.

MV algebras are a generalisation of Boolean algebras and provide the semantics for Lukasiewicz logic, an infinite valued logic. There is a close relationship between MV algebras and Riesz spaces since every MV algebra can be embedded in a Riesz-MV algebra.

An MV algebra has operations satisfying:

- $(x \oplus y) \oplus z = x \oplus (y \oplus z)$,
- $x \oplus 0 = x$,
- $x \oplus y = y \oplus x$,
- $\neg\neg x = x$,

- $x \oplus \neg 0 = \neg 0$, and
- $\neg(\neg x \oplus y) \oplus y = \neg(\neg y \oplus x) \oplus x$.

We can then define the familiar operators $\vee$ and $\wedge$ as

- $x \vee y = x \oplus \neg(x \oplus \neg y)$
- $x \wedge y = \neg(\neg x \vee \neg y)$

The prototypical example of an MV algebra is the interval $[0, 1]$ together with operations $x \oplus y = \min(x + y, 1)$ and $\neg x = 1 - x$, however this can easily be generalised to multidimensional spaces $[0, 1]^n$, in which $\vee$ and $\wedge$ correspond to the component-wise vector lattice operations.

Thus we can adapt our learning technique to give us vectors which can be considered as elements of an MV algebra simply by ensuring that they are positive, and normalising them so that every component is between 0 and 1. This can be done since each component can be normalised independently without affecting the ordering.

We can then use the standard semantics of Keenan and Faltz, but with MV algebras instead of Boolean algebras. I think it is likely that all the properties of Boolean algebras that they use are present in MV algebras, but we would need to confirm this.