Publication and Researchers:

The second paper I have chosen to study is 'The Sum of Its Parts': Joint Learning of Word and Phrase Representation with Autoencoders. Their are two researchers who are a Ph.D student and supervisor. The supervisor is Ronan Collobert who now works for Facebook at Menlo Park. Remi Lebret was a 4th year Ph.D candidate in electrical engineering. His interests in AI and Natural Language Processing (NLP) led him to building models for large datasets which led him into deep-learning models.

Remi has published the following papers in NLP:

- Simple Image Description Generator via a Linear Phrase-Based Model.
- N-gram-Based Low-Dimensional Representation for Document Classification.
- Rehabilitation of Count-based Models for Word Vector Representations.
- Word Embeddings through Hellinger PCA,
- Is Deep Learning Really Necessary for Word Embeddings?

As the paper states 'Human language “makes infinite use of finite means” '. Because there are finite words but infinte sentence for NLP to work there must be some effort to understand not only word meanings but also how they combine. The modern trend appears to be to create a word representation and to figure out a way to combine them for phrase representation. Many techniques have been tried. Simple one such as adding and multiplying have been competitive with techniques from Logic, Category Theory, and Quantum Mechanics.

This paper tries to combine the distributed representations via an autoencoder which is a type of deep-learning model that tries to recreate the input that it received.

Dataset:

The English corpus of words on Wikipedia as lower case words and with any sequence of digits replaced with 'NUMBER'. This is then parsed by the Stanford Tokenizer leaving 191,268 words.

Only the 10,000 most frequent of these were used for calculating co-occurrence probabilities. A symmetric 10-word window around each word used the following formula that to me looks like Bayes' Theorem:

$$p(c|w) = \frac{p(c,w)}{p(w)} = \frac{n(c,w)}{\sum_{c_j \in \mathcal{D}} n(c_j,w)},$$

The data was then encoded in a 100-dimension vector.

The authors also introduced a novel tak for evaluating phrase representation. This required a dataset of English-language phrases. To do this they extracted 5,000 phrases from wikipedia. 2244, 2050, and 547 were 2,3, and 4 word phrases respectively. The remaining 179 ranged from 5 to 8 words.

Findings:

5 tests were used for comparison with 4 other systems. The test are different ways to test word-similarity knowledge. For example one test consisted of 8,869 semantic questions such as 'Bern is to Switzerland as Paris is to ..?' or 'uncle is to aunt as boy is to ..?'

The table on the right seperates the results for Singular-Value Decomposition and the author's model as these both use techniques to reduce the dimensionality of data. As such the autoencoder approach performs significantly better on each test.

	WS	RG	RW	SYN.	SEM.
CBOW	0.57	0.47	0.32	53.5	22.7
Skip-gram	**0.62**	0.49	**0.39**	66.7	53.8
GloVe	0.56	**0.50**	0.36	**79.7**	**75.0**
SVD	0.43	0.39	0.27	52.3	34.1
Our model	**0.62**	0.49	**0.39**	69.4	43.0

The phrase-evaluation task consisted of feeding a phrase to the model and finding the individual words. For each phrase the model would return a ranked list the most likely combinations of words. The success of the model was measured using recall @ k. In the table on the right we can see that over 64% of the time the model produced the correct words at the top of the list.

	R@1	R@5	R@10
CBOW	11.33	29.56	38.46
Skip-gram	7.96	22.26	30.04
GloVe	54.97	79.97	86.54
SVD	17.42	32.87	40.72
Our model	**64.22**	**91.72**	**95.85**

One thing to watch is that the phrases were chosen from the same source that the co-occurrence statistics were calculated from. When you have people like Bryan Henderson making 47,000+ edits to correct the phrase 'comporting of' [1] you may have something that may not generalise to a lesser standard of English.

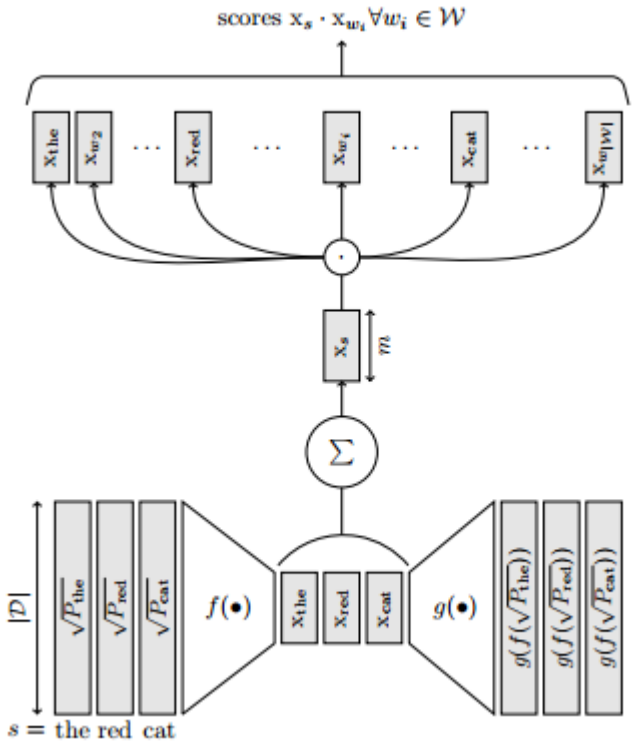
Techniques:

For a project like this to work an efficient representation of the words is needed. This involves reducing the vectors to a lower dimensional space. Traditionally this has been done by Principal Component Analysis but an autoencoder was trained to do the job here.

Starting in the bottom left the square root of the values (its co-occurrence probability distribution) of each of the wrods 'the', 'red', and 'cat' are fed into the autoencoder and encoded in a lower dimension.

A decoder is then trained to reconstruct the intial inputs. This is the first objective function.

The second objective function is:



$$\sum_{s \in \mathcal{S}} \sum_{w_t \in s} \sum_{\substack{w_i \in \mathcal{W} \\ w_i \notin s}} \max(0, 1 - x_s \cdot x_{w_t} + x_s \cdot x_{w_i})$$

I'm not sure where dot product fits in with BEMDAS but the aim here is to place words that appear in similar contexts together. This works because synonyms should have similar co-occurences and the values of phrases are achieved by simple element-wise addition.

Relation To Work:

One idea I have for the final project is to write a program to help people learning a language find articles to read that are at their level. From my own experience of trying to learn French you start by learning words and then you learn some grammar rules for combining these words and then there is a large gap to where you have to handle real-life examples.

One problem I noted was that you could read something and mostly following the gist of it but then come to a paragraph or paragraphs that was outside the scope of your ability. The problem is frustrating because your ability does not grow with constantly in all directions. For example you do not know 50% of the words related to sport and 50% related to food and then find that 6 months later you know 60% of each. Instead your knowledge grows as a function of what you are exposed to.

One solution I tried was to create flashcards from a source. I started with «Harry Potter et le coupe de feu» and ran each sentence in the first chapter through Google Translate and created flahcards for the Ankidroid program. The problem that emerged was that this very clearly gives the story away. Another solution might be to take each word individually but then synomony would come into play.

The solution above used with autoencoders could provide one part of the puzzle (another being a ranking of the difficulty of words). I am picturing something similar to a chrome plugin for translation that would keep track of the words and phrases that a user has looked up to get a sense of where in space they are found and how difficult they are (I'm also realising now that parts-of-speech tagging will be neccessary as in phrases like dark-blue jumper, light-red jumper, ugly green jumper there it might not be possible to guess the adjectives from context).

References:

[1] Obsessed Wikipedia Editor http://uk.businessinsider.com/wikipedia-comprised-of-bryan-henderson-wikignome-2015-2