UNIVERSITY OF LEEDS

Scaling to big data

Professor Eric Atwell: Hello, this is Eric Atwell and I'm going to, in this lecture, look at three research papers and summarise some of the main findings. And these are research papers in the area of word embeddings and scaling to big data. So in these papers-- and there's a Google code archive as well, to go along with them-- you'll learn more about methods and software to learn these word embeddings.

You'll be introduced to word embeddings in the speech and language processing textbook, the chapter on representing word meanings. And word embedding is the standard term for a numerical vector representation of word meaning, derived from a corpus. By looking at the examples of the word, the context that it appears in-- and the set of contexts represent meaning-- and then you capture the set of contexts in a vector. And we also look at the impact of scaling from small data sets to large, real-world data sets.

A lot of early research was on small data sets. Probably the coursework exercises you did, for example, in the data science module had a relatively small data set to learn from. But for applying real systems like Google Translate, for example, you have to have very large data sets. There's three papers. First of all, a very early paper showing you how difficult it was to do anything at all back in 1986 in this field. And then, later on, 2001, Michelle Banko and Eric Brill from Microsoft Research did a number of experiments on different machine learning algorithms to see how they scaled from very small to very large corpora. And then in 2013, this is a seminal paper, a groundbreaking paper by Thomas McAuliffe and his group from Google Research on efficient estimation of word representations in vector space. The real problem up to this point had been it took a lot of processing. And they found a faster way to build these vector representations. And you can try out the Google code yourself. So a bit of background before we start on these three.

First of all, I talked about AI research papers. Just in case you didn't realise, a lot of AI research, not just in universities but also in industry research labs-- Google Labs, Microsoft Labs, Facebook Labs,

and so on-- is reported. And what happens is that researchers do some experiments, find some new knowledge, and then write it up in a paper. And this is peer reviewed. Other experts will read it and make sure it is right and good. And then, if they accept it, then it's usually presented at a conference. You give a presentation, a bit like this lecture, and then you write it up in a conference paper for the proceedings. And for a conference, the paper may be four pages to eight pages or maybe slightly longer.

If it's a more substantial piece of research, then it can be submitted to a journal. A journal doesn't go alongside a conference-- usually it's just a separate thing that is published and contains longer papers with more details in it. So for example, as I mentioned previously, the ACL-- the Association for Computational Linguistics-- they have a website with lots of information to do with computational linguistics, including a page with listing computational linguistics journals. Not just the Computational Linguistics journal that they publish, but also other journals in this area. And also, a list of conferences in the field. And you can attend some of these online or in reality, after COVID is over.

And they also collect papers from the proceedings and from the journals-- mainly from the proceedings. And there are also these things called workshops. So a major ACL conference may have 1,000 people attending it and, as well as the main conference, they also have workshops for special interest groups. And they have smaller collections of papers. All these papers are collected in the ACL Anthology, which you can search and find. So if you want to find out anything on text analytics, it's often there. Of course, there are other journals which are not included, other conferences which are not included-- but they're fairly comprehensive.

One other thing to note as we proceed is the word corpus is based on the Latin word "corpus", which means a body. So if you Google corpus, you may find medical websites where they talk about corpus, because medical terms are often borrowed from Latin as well. Latin is the language of the Roman Empire from 2000 years ago. Well, they actually carried on up until 1,000 years ago. And then was used by the Catholic Church and various other organisations for scientific documents. And the plural in Latin, of corpus, is corpora-- or "core-pore-ah", depending on how you pronounce it. So I will be using the word "core-pore-ah." English speakers sometimes use the plural corpuses as the English type of plural for corpus.

Note also that the word proceedings is a sort of singular or plural. And I invite you to check this out for yourself in Google or in Sketch Engine. So there is often a conference proceedings listing the papers of the conference. So in some sense it is singular, but it has many papers in it so it's sort of plural.

OK - that's the background. And you might want to ask why am I reading these AI research papers? Or why has Eric Atwell asked me to read these things? Well, one is obviously to learn about AI research. So if you want to find out about the current research, you can google things like word embeddings and you'll find websites which may have a tutorial discussion about it. But if you want to find something which has been latest research, which has been peer reviewed-- other experts have read it and checked it, and maybe even asked for corrections or changes before it's published. So, to keep up to date.

The other thing is, I've been asking you, for coursework, to write a research proposal. And how do you think of a research idea? Well, one thing you could do is the research papers were often similar in structure to the research proposal. So one thing could do is, if you find a research paper that you like, you could end up with writing a proposal, which is similar, but maybe doing on a different data set or for a different company or for a different user group. I've suggested this before.

Your research very often is related to some previous research but changing some of the parameters-- like changing using it for different language, for a different user group, or for a different but similar sort of task. So a typical research paper, somewhere at the beginning, possibly even the title, it has the aim of the research. And then there's usually some background-- what other people have done similar to this before. And then you typically have methods explaining what is the program or research you're doing, which is what you need for your proposal. Except, in a paper, typically they don't give durations of work packages. They just say what they do. And then there's, in addition, some results. This is something you don't often get in a proposal because you haven't done the research yet. But in a research paper, the whole point is to present some interesting, new results and the conclusions that you've drawn from these results, which is the contribution to knowledge and why it's important, why it's worth other people reading this. So most of the sections in a proposal will also appear in a research paper. So reading research papers gives you some good ideas about what a proposal should look like.

Finally, let me know-- I did say proposals don't usually have results. One way of adding to a research proposal is to do some pilot study or proof of concept study. So you can implement something simple and then give some initial results. And that makes your research proposal more plausible because it shows that your ideas do work, at least on a small scale. That's just an idea for you to think about.

We've dawdled enough now. Let's get on to these actual papers.

So the first one-- I'm Eric Atwell, I get to choose the papers-- is one of my papers. This is actually a very early paper, and it more or less illustrates how difficult this is. How hard it is to do word embeddings. So the aim of the paper "A Parsing Expert System which learns from Corpus Analysis." This was presented at a corpus linguistic conference, the ICAME conference-- the International Computer Archive of Modern English conference, in 1986. And the aim of the paper was a discovery procedure for grammar. A way of taking a corpus and somehow automating a learning model of what words go together and why. And the background of this is it discusses other work on English grammar on part of speech taggers. And, in particular, on bigram models or Markov models that you saw earlier.

And the methods used were, essentially, to take a small corpus-- well, at the time 200,000 words or 1/5 of the Lancaster-Oslo-Bergen corpus of a million words. So it's quite small but that's all that the computers could cope with at the time. And for each, just choose the words which occur 100 times or more. And for those words, get a list of left contexts and right context-- the words that appear to a left before, and words that appear to the right after. And for all possible pairings of one word with another word, you compare W1 and W2. If the context lists of the first word were significantly similar to the context lists of the second word, then the two were deemed to belong to the same word class. So this is like if the embedding of word one is very similar to the embedding of word two, then merge them together into one joint word class. So the conclusion, well, this is the first corpus linguistics approach to learning word embeddings from a corpus.

There are also psychologists trying to do it to mimic human brains using early neural nets at the same time. This isn't using neural networks but just using basic statistics. But it was very limited. That the

processes at the time could only cope with 200,000 words. That was the biggest memory available. Not the whole of a corpus could go into memory at once. And only 175 words occurred more than 100 times, so only the most frequent words could be dealt with. And even then, the run took-- it reports several weeks on the University mainframe computer, dedicated just to running this thing. You could look in the paper for some details of the results.

We see here for each-- first of all, the threshold was set quite high at 0.8. And then words which were very similar were merged together. So the word "in" and the word "for" is the first pair, the most similar pair. And they were merged into one. And then the word "is" and the word "was" were found to be very similar in terms of context they appear in and, therefore, merged into one group class, and so on. So "will," "should," "could," "must," "may," and "might" were all merged into a single word class on the basis of the immediate lexical context the words that appeared before and after them in the hundred or more cases. The trouble is, it was only a small number of cases. And of these words, 175 words, almost all of them were functional words. Very few of them were content words which actually have meaning. So we see in the list, for example, world and country were very similar in their embeddings, or year and week were very similar. But there's not that many examples. So there's not a lot of results, and you evaluate it by just taking it, noticing a couple of the examples, noticing groups of function words like world, should, and so on and also a couple of content pairs which were merged together.

OK. So by the year 2000, 2001, it was possible to deal with much bigger data sets. And Michelle Banko and Eric Brill worked at Microsoft Research Labs, so they had very powerful computers, much better, much more powerful than was available at universities. And this has tended to be more and more the case. But a lot of the very large scale machine learning research or deep learning research, the novel findings are happening at Amazon, Microsoft, Google, of course IBM, the big companies which are very large research labs and have very powerful huge computers.

OK. Banko and Brill, what they wanted to do was evaluate the performance of different learning methods trained on orders of magnitude more label data. So they really wanted to see what happens if you have more data, and what effect does it have on machine learning. And they chose as a particular task confusion set disambiguation. So for example, in a text the word pronounced weather

it could be either spelt W-E-A-T-H-E-R in context, like it is sunny weather today, or W-H-E-T-H-E-R in context, like I don't know whether to cry or laugh. So the nice thing about this problem is that labelled training data is essentially free. You can go through a corpus, and whenever you come across either of these words, the actual class is what is actually spelt, but the possibilities are the other. The one that is spelt and the other one, too. So you basically could go through the text. And if you come across the weather is sunny today, you replace that with the unknown is sunny today. And unknown has class either rain weather or the whether if weather. So the correct class is the one that the writer used to assume that in writing, the actual text that's found is the correct class.

But the other alternative class is from a dictionary. The other is possible spelling. So the method that they used to evaluate, they evaluated a range of different classifiers on a range of different data set sizes and came up with graphs showing what happens as you increase the training set, what happens as you try different classifiers. And we'll look at this in a minute in more detail. The overall conclusion, as you might expect, the more data you get, the better the systems are, the higher the accuracy. Furthermore, if you give more data, all the classifiers improve to the point where even the worst classifier for a small amount of data is much better if you give it more data. So that's the sort of overall result.

And the actual practical conclusion was that maybe text analytics researchers should reconsider the trade-off between spending time and money on algorithm development versus spending it on corpus development. In other words, that different classifiers is, why bother developing yet more machine learning algorithms when actually, all you need to do is take the ones you've already got and give more data? If you give more training data, that has a much better effect than tweaking the algorithm. So in your machine learning course or data science course, you may have tried several different learning algorithms to see which one was best, but you tried it on a set size of data. If you had tried on a much bigger set of data, you probably would have got better results. So researchers should spend more time collecting a corpus and labelling the corpus.

OK. Here's some actual results. This shows figure one, learning curves for confusion set disambiguation. So the data along the bottom axis goes from naught 0.1 million words up to 1,000 million words, or a billion American words. Actually, it started off with about 200,000 words. Notice it's

a logarithmic scale. So they tried it for about 200,000 words, and then a million words, and then 10 million words, and 100 million words, and then thousand million words. And the accuracy went up. They tried four different classifiers, and for all four, the accuracy increased. The worst accuracy was about 75%. The best accuracy rose up to about 97%.

Notice also that for a small amount of data, the four classifiers are ranked such that the Winnow-- that's the blue line-- the Winnow classifier is the worst for a small amount of data. For a much larger amount of data, the Winnow one was the winner. So Winnow, given more data, changes from being the worst classifier to the best classifier. On the other hand, the memory-based or case-based classifier, for a small amount of data, was the best classifier. Gradually over time, if you give it more and more data, turns out to be the worst classifier. So in general, which is the best classifier doesn't so much depend on the algorithm, but on how much training data you've got. You should forget about choosing the best algorithm or the best parameter settings for an algorithm. Just give more data, and they'll all get better. This is a bit disheartening for machine learning researchers, who really like tweaking algorithms. But actually, it's a waste of time. For computational linguistics, what really matters is you have a huge corpus, and if you pile up more and more training data, then your accuracy will improve regardless of which classifier you're using.

However, although we said training data for this particular task is free because you can take any amount of text and just replace whether with the ambiguous two cases, there are other problems. The larger the data set, the more, the larger the model that you're learning. So here we have, for the Winnow algorithm and the memory-based algorithm, the size of the internal model, that is, how much processor and memory you need to store the model and to compute with the model. And it grows and grows. And in this case, both of these axes are logarithmic. So as we see, as you get from one million words up to 1,000 million words, then the size of the memory that's required and the processor requires, also grows. But they haven't actually specified what the numbers are on the scale. But that's the general thing, that you need more memory, and you need a lot more processor power to deal with these much bigger corpora. And maybe that's not a problem for Microsoft or Google research labs. But for university researchers, it is a problem. You will find, for example, for Sketch Engine, you're limited to having a million-word corpus. If you try to go over, you just won't be able to. And if you want to do another corpus, you have to delete and remove the corpus you've got. And don't bother trying to

ask for more because we can't give you more. Well, maybe for your project later on. But for coursework exercises, you are restricted to memory and processor.

Again, another thing they looked at is they were looking at comparing different machine learning algorithms over different data sizes. Well, a clever thing to do in machine learning is, if you're not sure which classifier is best, to have an ensemble. An ensemble is to use all of them or several different algorithms, and then just vote. If you've got five classifiers, let's say, and three of them say the answer is x and two of them say the answer is y, then voting overall, the answer is x. And they found that for small amounts of data and for medium sizes of data, that works very well. Once you get to very large corpora of 1,000 million words, then the best classifier can outperform the voting classifier. The problem is you still don't know in advance which one is going to be the best. You still have to try all of them to get the best. So the ensemble is generally better, even for large data sets, unless you have a very large training set and you know, for that large training set, which one is best. But in general, an ensemble is better than an individual classifier. OK. So that's a nice problem or task because you can generate any amount of training data just by replacing every ambiguous word with its other possibilities. Whenever you come across whether, you assume that there's two possible classes rather than just what's actually written down.

For real problems, that is not often the case. So you haven't got that. So what they did is they looked at what can you do in those sorts of cases, and they came up with two possible solutions, or at least strategies. One is called active learning. That is, if you've got an ensemble, several different classifiers, trained on a small amount of data, and you carry on using those to label a large amount of data, and you notice which of those are the most uncertain instances. So if it turns out that three of them vote one way and two of them vote another, that means the classifiers are not sure. Therefore, that particular instance should be given a label by hand. So active learning means choosing carefully which instances, you're going to bother labelling because labelling costs if you're going to get humans involved. You only label the instances which the classifiers aren't so sure about. And then that way, the classifiers learn from the difficult cases.

The other thing I suggest is semi-supervised learning. Choose it in instances which have the highest probability of being correct for automatic labelling. So if you have an automatic classifier trained on a

small data set, and then you continue to use it on a growing data set, then it will choose-- most classifiers, as well as giving a label, they will give some probability that the label is correct. And if you only use the ones where the classifier is very sure-- in other words, for example, if all of the different classifiers in an ensemble, they all voted for the same, for x rather than y, then you're quite sure about that. The other conclusion they came is one we said before, that researchers should direct efforts towards increasing the size of annotated training collections and de-emphasise the focus on comparing different learning techniques trained only on a small training corpus, or only on small training corpora.

So for coursework, you may be given a small corpus and told, try out which classifier is best. But for real life, that isn't a sensible thing to do. What you should be doing is collecting a large training corpus and annotating a large training corpus for your particular task, and then use whichever classifier, or use some ensemble of classifiers and it will probably be OK.

OK. Let's go on to the third paper I wanted to read. And this is Thomas Mikolov and his group at Google Research Labs another 10 plus years later, 2013. They found that dealing with very large corpora of 1,000 million words, it's still-- even though you can use these for certain tasks like sense disambiguation, we just saw, it's difficult to learn very large word representations in vector space. So they wanted to come up with a novel model architecture for computing these vector representations of words for very large data sets. And as well as the model, they wanted to come up with a comprehensive test set for measuring these regularities, syntactic regularities and semantic regularities.

So as background, they noted that various other people had been working on neural network language models, feedforward and recurrent neural networks. And these do work. They do give you interesting vector representations. But they have very high complexity. They need a lot of processor, a lot of memory, and therefore take a lot of time. And in the paper, there's some more details of the orders of magnitude that's required. And then they came up with a much faster and much simpler method, in fact, two methods, one called Continuous Bag of Words or CBOW, and another continuous SKIP-GRAM, or just SKIP-GRAM. And these are much faster for training, for giving, for each word, a vector representing its meaning from the corpus. And because these re much faster,

they were able to do it on 1,000 million words rather than just a million words because they have much, much lower computational complexity than feedforward or recurrent neural networks.

They also came up with this comprehensive test set. And the comprehensive test set, they demonstrated their own methods and other people's methods, and showed, not surprisingly, that other people's methods had very low scores compared to their scores, which were much better. And they gave this test set to the research community for other people to try. Here we have a diagram, figure one, showing the model architecture. If you want to see more details, then I advise you to read the paper. But essentially, the CBOW model predicts the current word based on the previous, or rather the contexts. So wt minus 1 is the word before, and the word two spaces before and the word after, and the word two after. And if you take all of that context into account and somehow summate those for a very large number of examples, then you have a meaning of the current word based on the contexts.

And the other model is the SKIP-GRAM model. This predicts the surrounding words given the current word. So for the meaning of a current word is looking at various contexts of the word comes in, how good is the current word at predicting its context? The word w minus word, once before and two before and the word after and two after. And if you want to play with software, you can find various parameters you can change. Doesn't just have to be two words on either side. It can be more. And some of the other parameters can also be changed.

OK. So they came up with these models, which you can download and are very widely used as a way of generating vector representations of the meanings of words. And they did start off in the background saying other people had done this, too, but using much more complicated models which took a lot longer. The previous work, to compare the quality of different versions of word vectors-- previous papers typically use a table showing some example words and their most similar words. And then they say, well, these are right. We understand them intuitively. Trouble with that is, it sort of looks good to me as a way of evaluating. Here's some examples. They look good. That's it. So what they wanted to do, Mikolov, is to define a comprehensive test set containing five types of semantic relations or questions, and nine types of syntactic relations or questions. We've seen what they mean

by questions. But the words are semantically related in different ways. And in a minute we'll look at two examples of each category in the table one.

Overall, there's nearly 10,000, just over 10,000, nearly 10,000 semantic pairs, and over 10,000 syntactic pairs. So the question seems to be--and furthermore, in their tests they were very precise. The question is assumed to be correctly answered only if the closest word to the vector computed using this method is exactly the same as the correct word in the question. So you had to get the right answer.

A synonym was it was counted as a mistake. So here we are some examples of relations. So we saw that synonyms are words which mean exactly the same. But actually, words can be related to other words in different ways. So for example, Athens is related to Greece in terms of being a capital city, or Oslo is related to Norway in terms of being a capital city. So we see lots of different relations, including currency, so that the kwanza is the currency in Angola. Or a city in a state. Illinois is a state. Chicago is a city in Illinois. And there are also gender relationships. So brother and sister are related. And they have the same meaning, but one is a male and the other one's the female. Those are semantic relations.

There's also syntactic relations. So apparent, as an adjective, has a corresponding adverb, apparently. Or they have opposites. So possibly, the opposite of possibly is impossibly. So opposite is a syntactic relationship, a grammatical relationship rather than just one of these semantic relationships. And we see other ones like that. So the reason they have more syntactic relations is because it's clearer to see exactly what-- these are more clearly defined, if you like, opposites and superlatives and present versus past and plural as opposed to singular. Those things are very well-defined, whereas coming up with lots of semantic relationships is a bit harder. OK. So here we have some examples.

And then they tried-- remember, they had these very large data sets. They tried out some of the other methods. And most of them didn't do too well. Mikolov even tried their own recursive neural network system, and it came up with a score of 8.6% accuracy on the semantics and 36.5%. OK. So a lot of

these other systems. And then they tried their own version, a more upgraded version, and finally their own CBOW and SKIP-GRAM models, and these were much better. So the SKIP-GRAM model in particular, for the semantic relationships, got half of them right. And for the syntactic relationships, it got over half of them right.

They also developed a neural network language model with a much bigger training set than had been done previously. But that took a long time to do the training. So six billion words, as opposed to the other ones, which were up to 1,000 million words. So what this also shows is, if you look at training words column, using very large training corpora for a neural network model does give you better results than the other ones. So even with neural networks, simple neural network models, which take a lot of processor time, but the larger corpora give you better results. But the SKIP-GRAM, overall, is much better.

So what were they actually testing? So here we have some examples. The idea is given France to Paris, so France is to Paris and Italy is to what? And what you actually do is take the vector for Paris, subtract the vector from France, add on the vector for Italy. What does that give you? And you find whatever the vector is, what is the vector which is most similar to that? And it should be Rome, right? So Paris minus France Plus Italy gives you a vector, and the vector should be most similar of all possible vectors. Rome should be the most similar. And we see this generally works. There are some strange examples. So for example, Microsoft is to Windows as Google is to what? Well, Android. Yeah, it's OK. What about Microsoft is to Windows as IBM is to what? Well, IBM is to Linux. That's very strange. Linux IBM does also use Windows, doesn't it? Or Microsoft is to Windows as Apple is to iPhone? Well, it's because Apple doesn't really have an operating system, except as they have the iPhone operating system. So it does mean something. Now some of these are just wrong, I guess, like Japan is to sushi as France is to tapas. Well, tapas is a food you get in Spain rather than France. But nevertheless, it's not too far out.

OK. So conclusion is, it is possible to train high quality word vectors using simple model architectures, simple compared to the popular neural network models, feedforward and recurrence. Even when you do the neural network module, you'll find out more about feedforward and recurrent deep learning models. But because of a much lower computational complexity, it's possible to compute very

accurate high dimensional word vectors from a much larger data set. So Banko and Brill have previously shown, whatever machine learning you're doing, if you give it much more data, it will be better. And that's why SKIP-GRAM was better because it can cope with much larger data sets. They also conclude that they have this comprehensive test set with lots of examples. And you don't just say, here's an example, it looks good so my system must be good. You actually have to test it and score. And the ones which previously seemed to be OK came up with quite bad percentage scores overall. So a test that will help the research community to improve. And overall, high quality word vectors will become an important building block for future NLP applications. Pretty much all NLP systems now-- we're going to look at machine translation and information retrieval and so on-- all of these systems now, they don't work on words, but they work on vector representations of words extracted using something like word2vec.

Finally, if you want to, you can go to the Google code archive and you can look at the word2vec tool. It takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data. It gets a list of the words. You can say you only want to have the most common words when you specify the parameter. It then learns vector representations for these words. The resulting word vector file can be used as features in many natural language processing and machine learning applications. So then you go on to do classification or whatever, using not the words, but the vectors of each words. A simple way to investigate the learned representations is to find the closest words for a user specified word said. So as well as the word2vec, it also has a number of tools such as distance to let you try this out. For example, you can enter France, and distance will display the most similar words and their distances to France, distances in terms of similarity of vector, not in terms of how far you are away from the real France. The Google code archive website also gives you pointers to where you can get large training corpora from lots of different sources, not just Google's own. And also, you can download the pre-trained vectors if you want to to use them yourself. If you want to, you can train your own vectors for your own corpus. That may take a lot of time and processing. Or you can just use Google's for themselves.

OK. So in summary, in these papers, you should learn about methods and software to learn word embeddings, that is, numerical vector representations of word meanings. I would say don't try the Atwell method yourself. I mean, it might be an interesting project to reproduce those results using

current technology rather than 1986 technology. You don't have to use the Pascal programming language or any of those obscure things from 1986. And you can see the impact on scaling from small data to large real world data sets.

So now I've told you the main findings, I recommend that you go away and read these for yourself to get a better understanding, and also to get an idea of what a research paper is like so you can see what a research proposal should be like. So the paper from 1986 by myself was a very limited first attempt to try to do this. And then onto 2001, Banko and Brill at Microsoft Research Labs showed all sorts of experiments that basically, the more data you have, whatever your classifier is, it will get better. More data gives higher accuracy for all classifiers. And Mikolov and his colleagues at Google Research Labs displayed the Continuous Bag of Words and the SKIP-GRAM models as two different algorithms for efficient estimation of word representations for very large corpora. And you can try at home their tools from the Google code archive.

OK. Go away, read the papers, and try this out yourselves. And enjoy. Thank you very much for listening.

[END]