

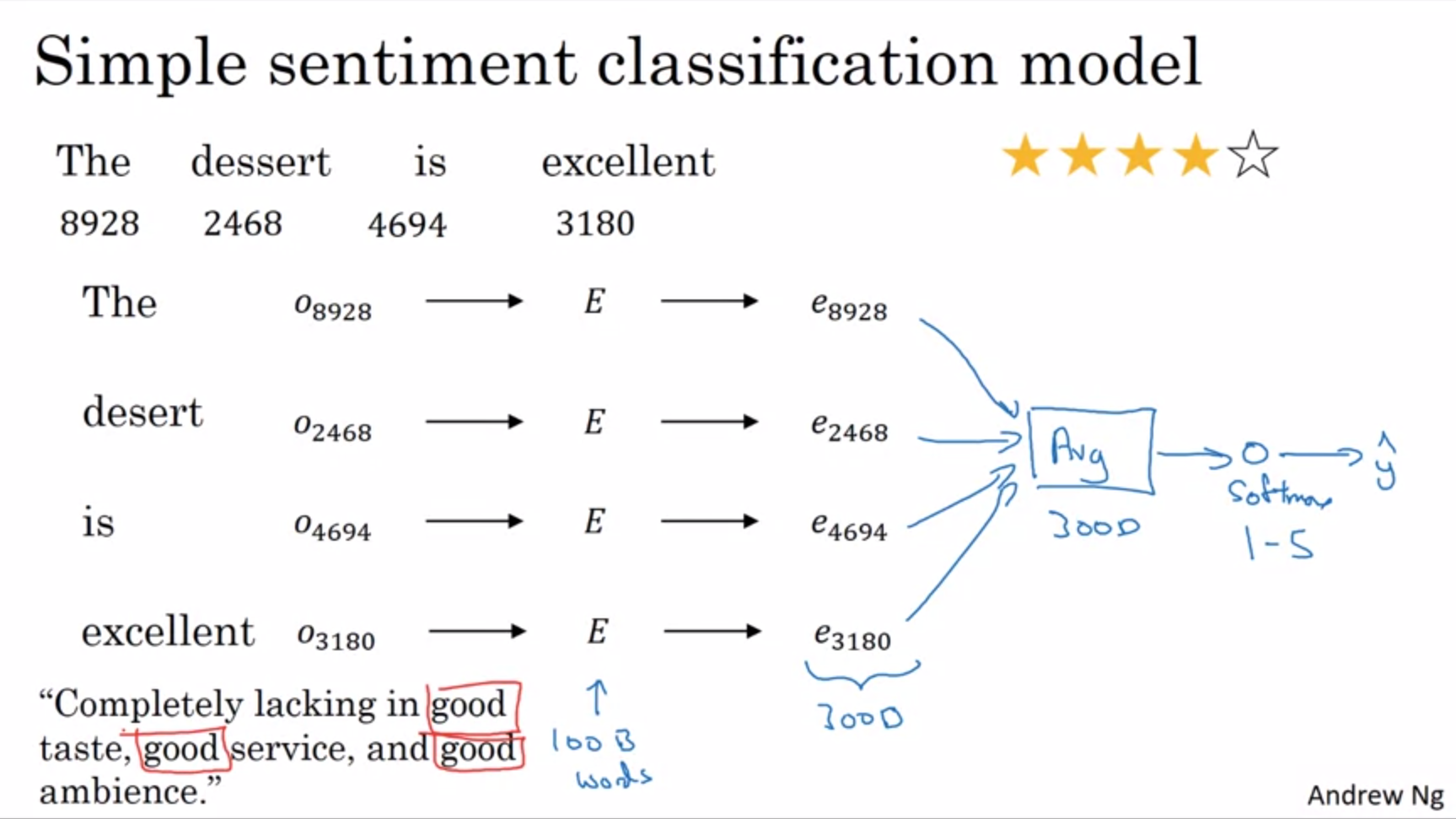
Sentiment classification is the task of looking at a piece of text and telling if someone likes or dislikes the thing that they are talking about. It is one of the most important building blocks of NLP. A challenge is often that the training set is not that big. We can help ourselves with word embeddings in those cases.



You can build a system to monitor comments, e.g. what people are saying about a restaurant on social media (twitter, facebook, Instagram). Like that you can keep track of problems and see whether your restaurant is getting better over time.

Word embeddings can help us to do much better on this training task, especially with small datasets.

We will go through a couple of algorithms.



We use our 10k vocabulary as usual.

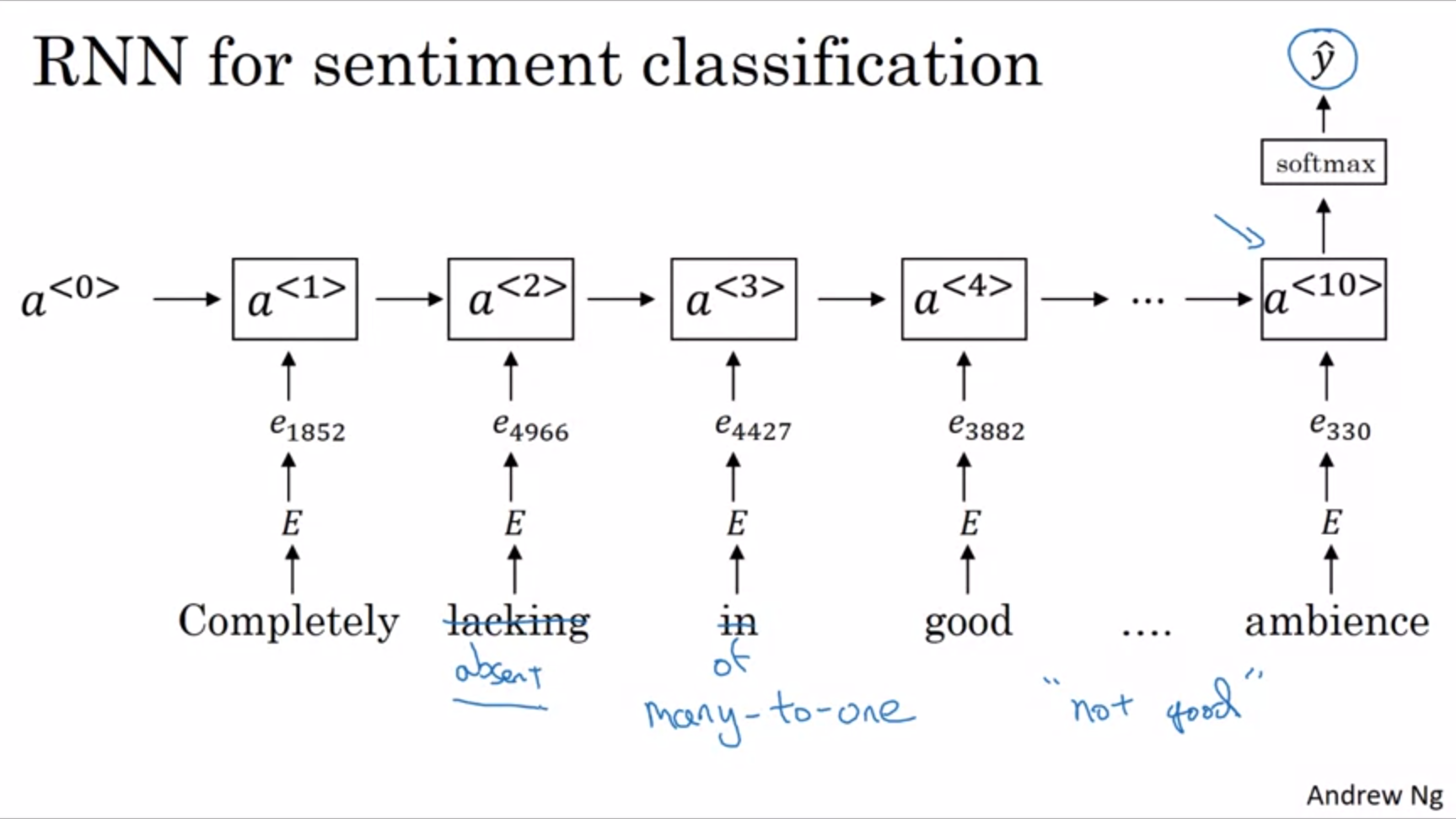
The embedding matrix E can be learned from a much larger text corpus (e.g. 100B words). This allows us to take a lot of knowledge even from infrequent words or words that were not in the training set.

We could just average or sum the word embeddings and pass the resulting 300D feature vector to a softmax unit with the 5 possible outcomes 1 star – 5 stars. This will work for short or long sequences.

This algorithm will work decently well. It averages the meaning of all the words in your sentence.

Problem of this simple algorithm: It does not consider the order of words. This will be problematic for the last sentence with 3x “good” in it.

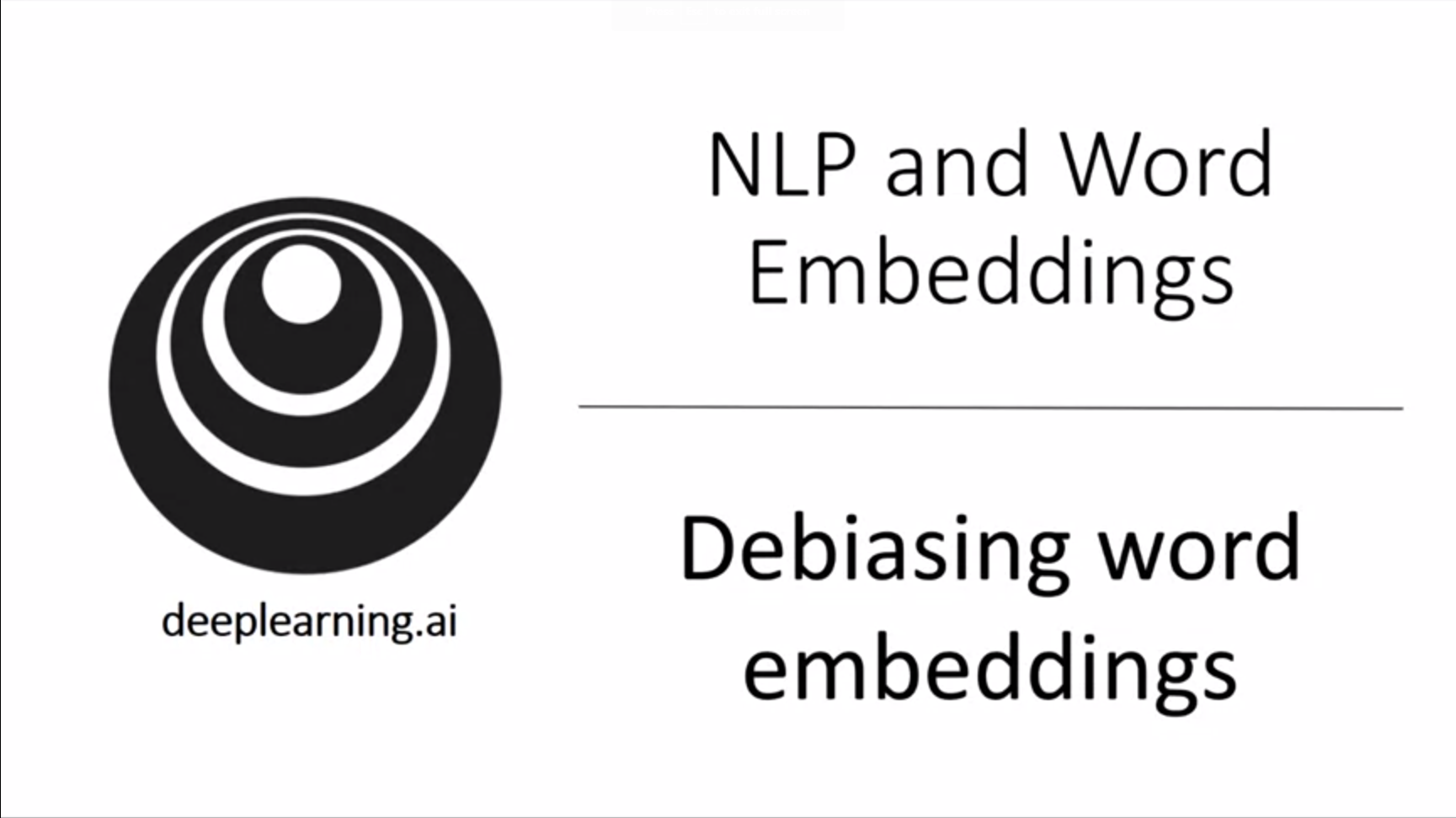
Instead of summing / averaging all word embeddings we can use a more sophisticated model, e.g. RNN.



Start with 1-hot vector. Feed embeddings into many-to-one RNN. This will be much better with taking into account sequences of words and will recognize that “lacking good ambience” is very different from “good ambience”.

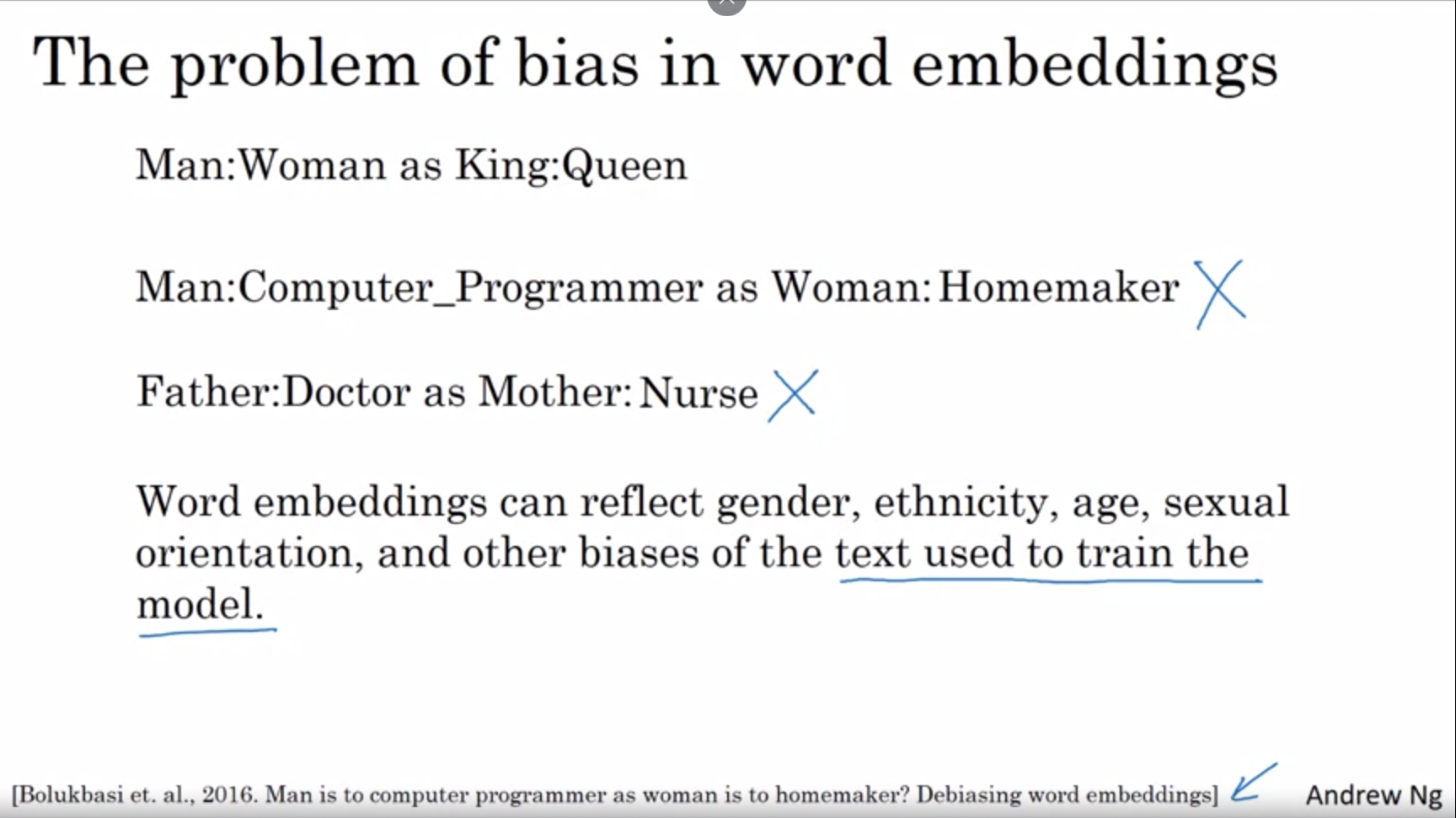
Using embeddings will do a much better job at generalizing (also to new words not seen in training set). E.g. even if absent is not in the training set of the sentiment classification problem, this will likely work with embeddings.

With pretrained word embeddings we can quite quickly build effective NLP systems.



ML algorithms are increasingly trusted with making important decisions. We need to make sure that there are no undesirable forms of bias such as gender bias or racial bias.

We will see how we can tackle the issue of bias in word embeddings.

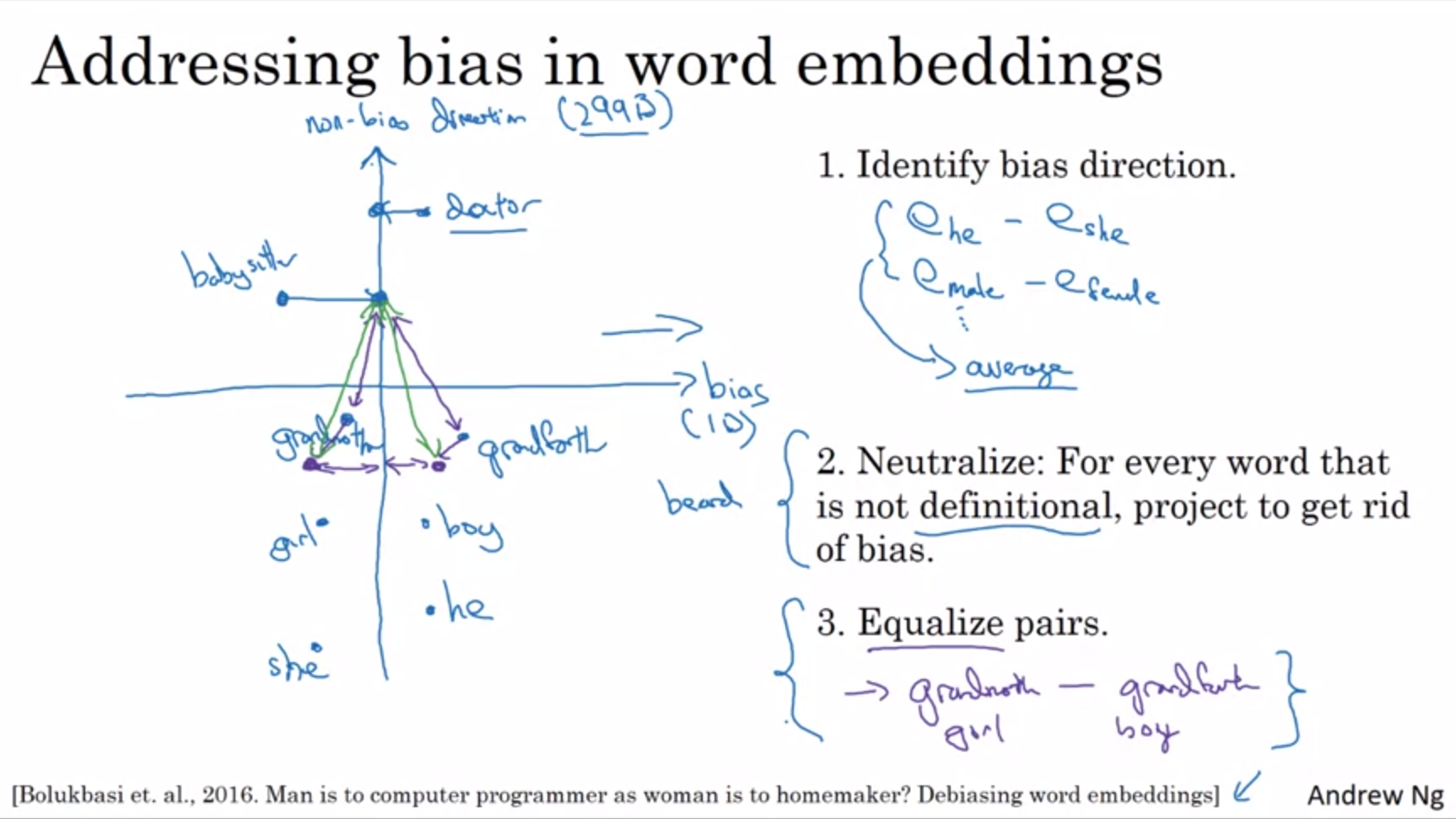


Second and third examples are an example for undesired gender bias. Also socioeconomic bias is quite common.

Nowadays ML makes a lot of important decisions in a human’s life such as for loan applications, whether people will get a job and for sentencing guidelines in the criminal justice system. It is essential to diminish as much as possible or eliminate biases.

The good thing is that we have better ideas how to quickly reduce the bias in AI than in the human race.

We will now look at one example of a set of ideas to reduce the bias in word embeddings.



Let’s say we have already learned a word embedding as displayed on the diagram on the slide.

1. Identify bias direction: calculate the average over a few differences between male and female words. Here the gender direction is the horizontal direction as marked by the bias arrow. The vertical direction is unrelated to the particular bias that we are trying to address. The non-bias direction is a 299D subspace whereas the bias direction is 1D. The bias direction can be more than 1D in which case we need to use SVD instead of simple difference. SVD uses similar ideas than PCA.
2. Neutralization step: get rid of bias. For some words it is ok to be gender-specific such as boy, he, grandfather. Other words we want to be gender-neutral such as doctor, babysitter (these are not definitional words). We will project these words on the vertical axis to eliminate the gender component.
3. Equalize pairs: for words such as grandmother and grandfather we want the only difference to be the gender. E.g. the difference between grandfather and babysitter and grandmother and babysitter should be the same. Otherwise this would enforce an unhealthy bias. In terms of linear algebra this means that the words should be equidistant to the vertical axis here. There are many words like this where we need to carry out equalization step.

How to decide what to neutralize? E.g. the word beard maybe should be closer to man than to female. What the authors did is to train a classifier to decide what words are definitional. It turns out that most words are not definitional and only a small subset of words is definitional and therefore should not be neutralized. The pairs that need to be equalized is generally also a small subset. It is feasible to handpick those pairs.

The full algorithm is a bit more complicated than presented here.

To summarize, reducing bias from algorithms is a very important problem since these algorithms are used to make more and more important decisions.

There are many algorithms. We have seen one. This is a very active area of research.