Embed, Encode, Attend, Predict

A four-step framework for understanding neural network approaches to Natural Language Understanding problems

Dr. Matthew Honnibal
Explosion AI

Taking a computer's-eye view of language

- Imagine you don't speak **Indonesian**. You're given:
  - ⭐ 10,000 rated restaurant reviews
  - 📝 a quiet room, a pencil and paper
  - 📆 one week
  - ☕ a lot of coffee
- How could you learn to **predict the ratings** for new reviews?

Siang ini saya mau makan di Teras dan tenyata penuh banget. Alhasil saya take away makanannya. Krn gak sabar nunggu. Benar kata orang kalau siang resto ini rame jadi menyiasatinya mungkin harus reserved dulu kali a

Dateng ke sini karna ngeliat dari trip advisor... dan ternyata wow... ternyata reviewnya benar...makanannya lezat dan enak enak... variasinya banyak..dan dessertnya...es kopyor...super syegeer..saya suka

Teras dharmawangsa tempatnya enak cozy place, enak untuk rame-rame, mau private juga ada, untuk makananya harganya terjangkau, terus rasanya enak, avocado coffe tidak terlalu manis

Machine Learning and the reductionist’s dilemma

- Machine Learning is all about **generalization**
- What information matters in this example, and what’s irrelevant?
- Most sentences are **unique**, so we can’t process them holistically.
- If we can’t reduce, we can’t understand.

How to understand reviews in a language you don't understand

- Do the words in it usually occur in **positive** reviews or **negative** reviews?
- Track a **positivity score** for each Indonesian word. When you see a new word, assume its positivity is `0.5` .
- Count up the **average** positivity score for the words in the review.

Bag-of-words Text Classification

- If `total > 0.5` and review is **positive**,
  or `total < 0.5` and review is **negative**
  Your theory worked! Next review.

- If `total > 0.5` but review is **negative**:
  Your positivity scores for these words were too high!
  Decrease those scores slightly.

- If `total < 0.5` but review is **positive**:
  Your positivity scores for these words were too low!
  Increase those scores slightly.

What are we discarding? What's the reduction?

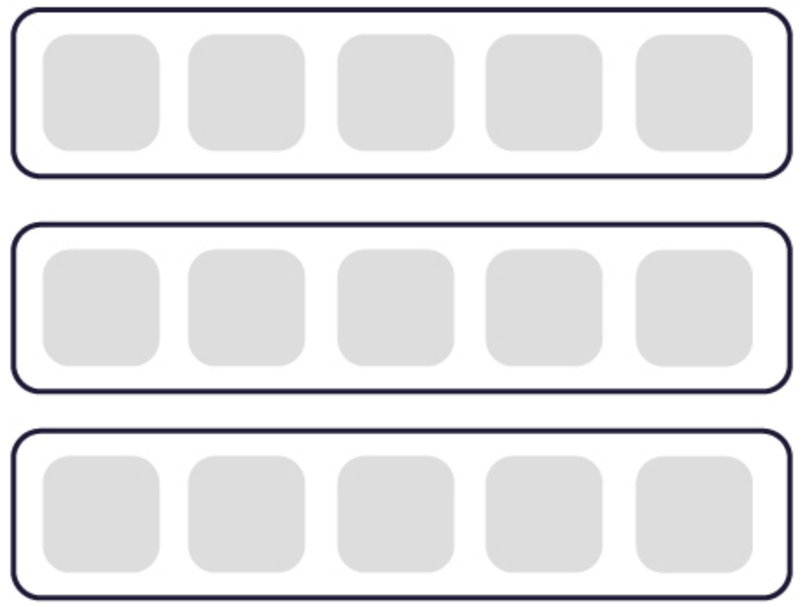
- We're assuming:
  - different words are unrelated
  - words only have one meaning
  - meanings can be understood in isolation
- How do we avoid **assuming** this?
- How do we learn what to learn?

Embed. Encode.
Attend. Predict.

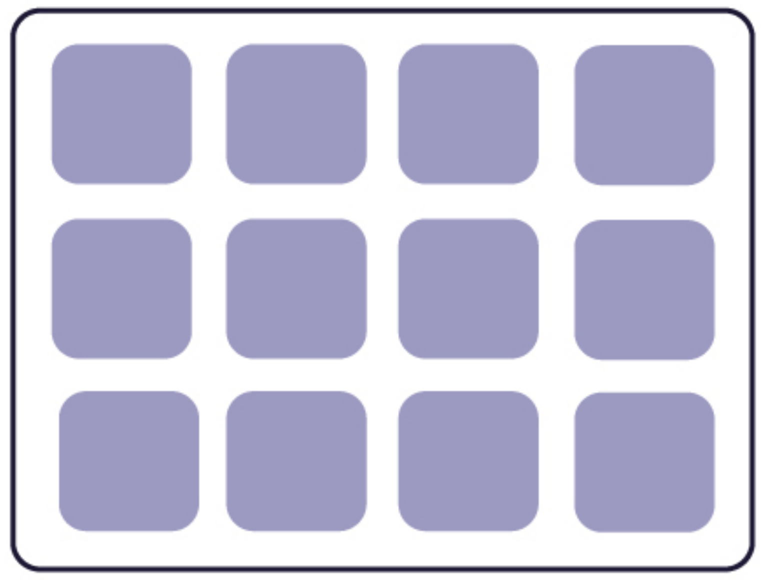
Think of data shapes, not application details.

integer
category label

vector
single meaning



sequence of vectors
multiple meanings



matrix
meanings in context

Problem #1

All words look unique to the computer

- “dog” and “puppy” are just **strings of letters**
- easy to learn `P(id | "dog")`
- need to predict `P(id | "puppy")`

Solution #1

Learn dense embeddings

- “You shall know a word by the company it keeps.”
- “If it barks like a dog…”
- word2vec, PMI, LSI, etc.

Problem #2

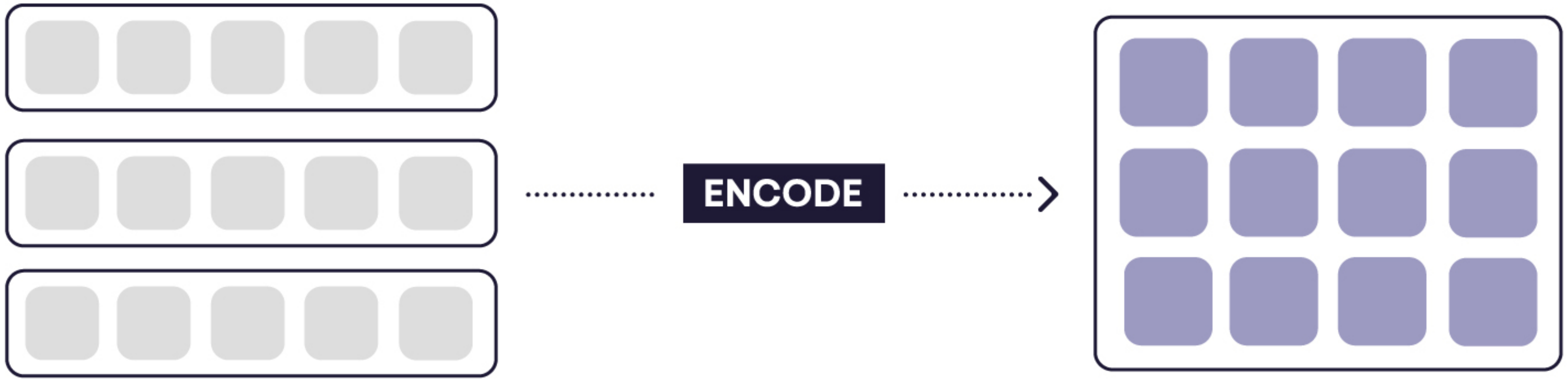
We’re discarding context

“I don't even like seafood, but the scallops were something else.”

“You should go somewhere else. Like, literally, anywhere else.”

Solution #2

Learn to encode context



- take a list of **word vectors**
- encode into **sentence matrix**

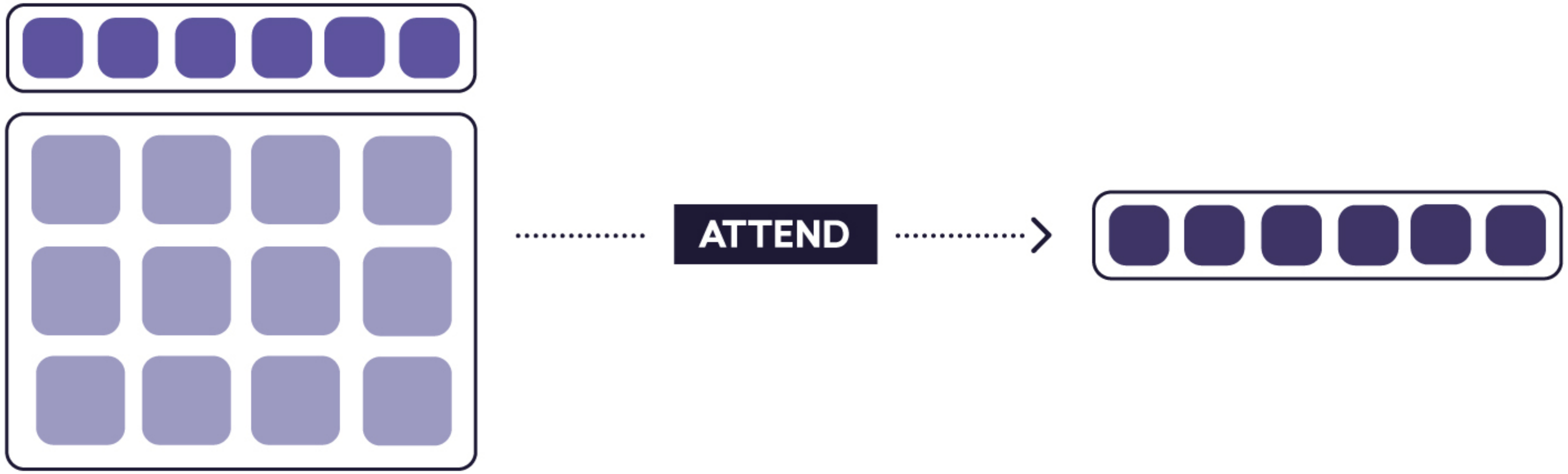
Problem #3

Too much information

- Okay, you’ve got a **sentence matrix**. Now what?
- rows show meaning of individual tokens
- no representation of entire sentence

Solution #3

Learn what to pay attention to



- summarize sentence with respect to query
- get global problem-specific representation

Problem #4

We need a specific value, not a generic representation

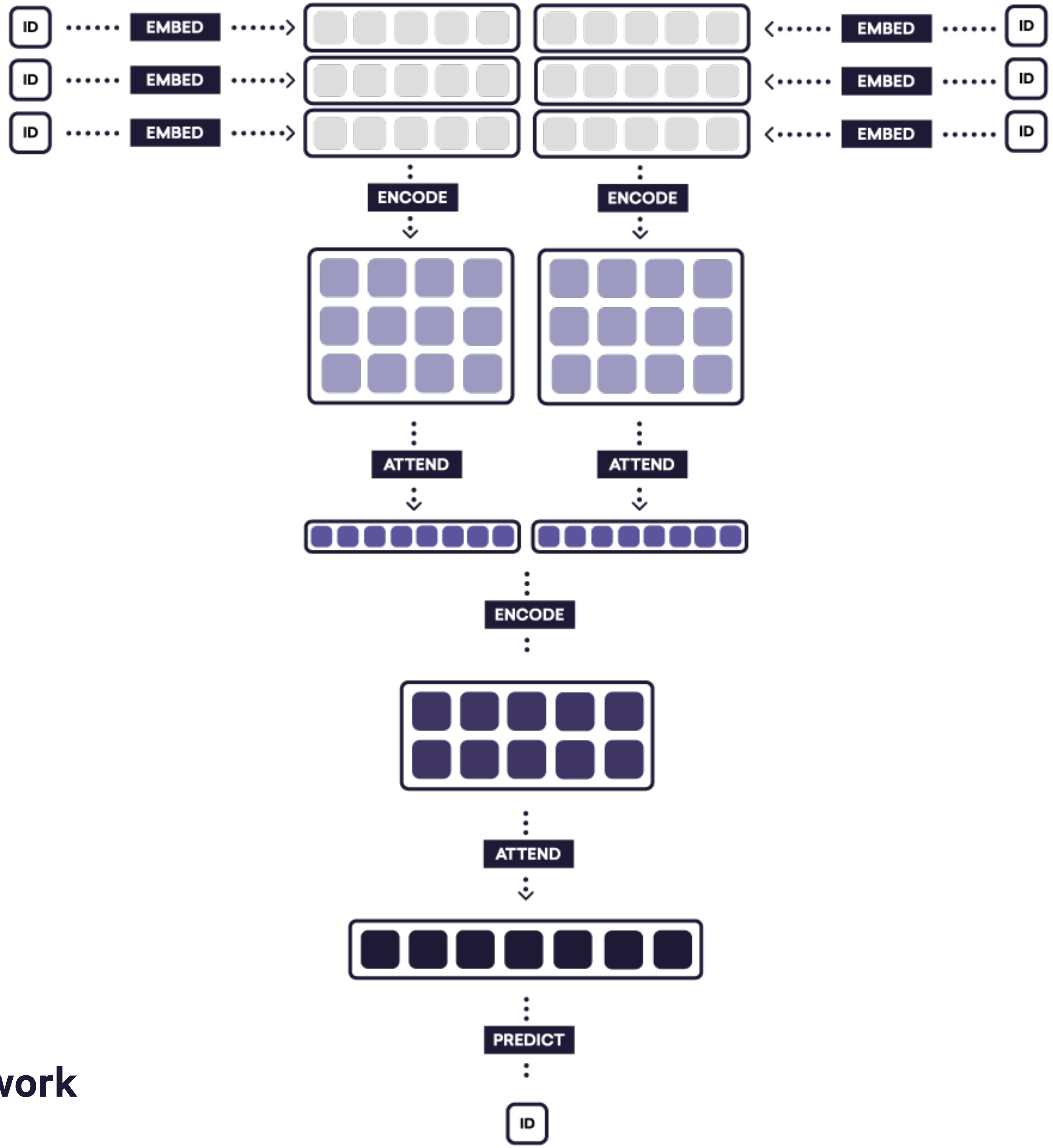
- Okay, you’ve got a **sentence vector**. Now what?
- still working with “representations”
- our application is looking for a **value**

Solution #4

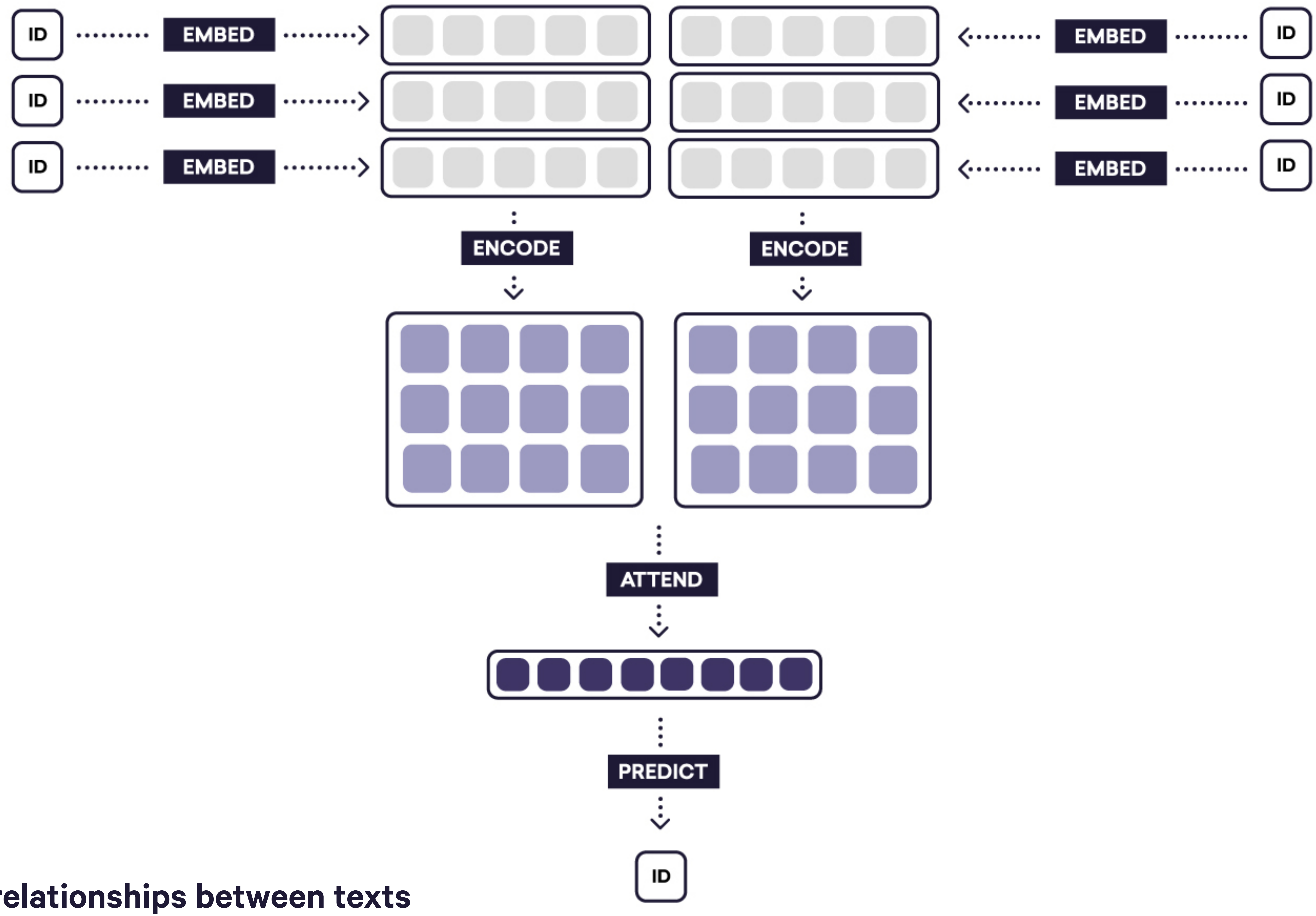
Learn to predict target values

- turn the generic architecture into a specific solution
- provide the value to your application

Putting it into practice



A hierarchical neural network model for classifying text



Predicting relationships between texts

What if we don't have 10,000 reviews?

- initialize the model with as much **knowledge** as possible: word embeddings, context embeddings, transfer learning
- save your data for **attend** and **predict**
- use general knowledge of the language for **embed** and **encode**

Conclusion

- neural networks let us **learn what to learn**
- knowledge must come from *somewhere*, ideally **unlabelled text** (e.g. word embeddings)
- you still need **labels** to predict what you're *really* interested in
- the **general shapes** are now well-understood – but there's lots to **mix and match**

Thanks!

💥 **Explosion AI**
explosion.ai

📲 **Follow us on Twitter**
@honnibal
@explosion_ai