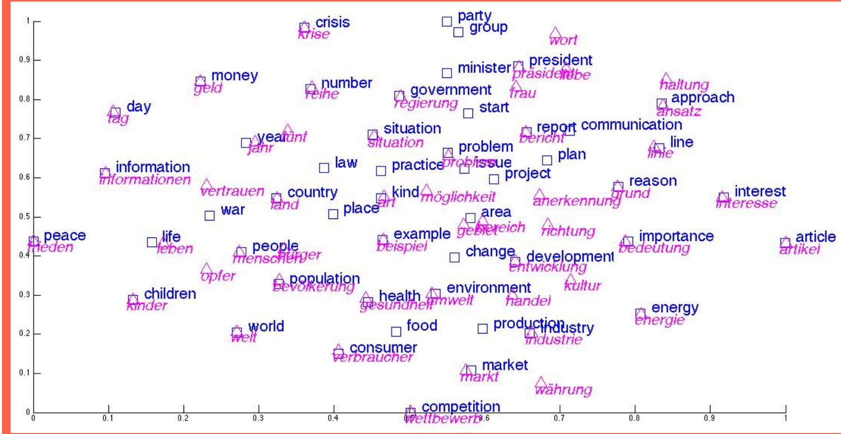
What is...machine learning in mathematics - part 3?

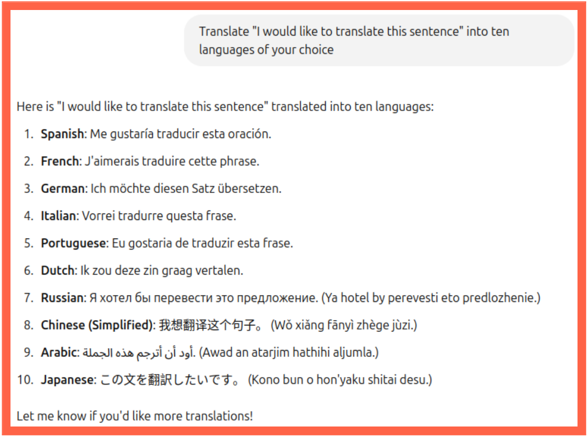
Or: Translate, please

Word embeddings



- The standard way to convert words into numbers are word embeddings
- Words in different languages end close after embedding
- This is essentially how dictionaries work

Meaning, please



Translate "I would like to translate this sentence" into ten languages of your choice

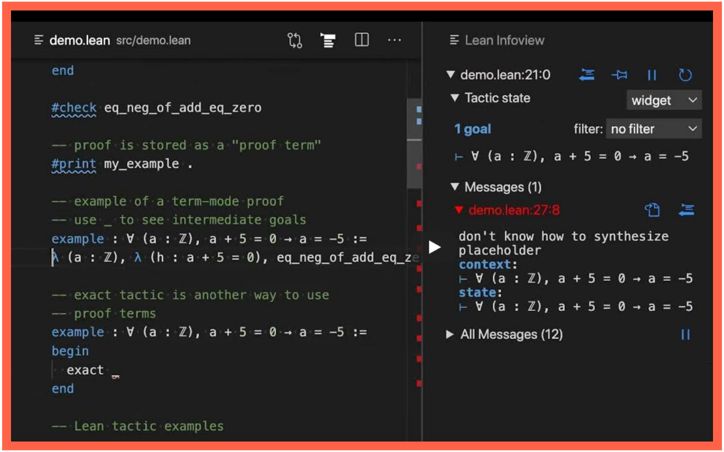
Here is "I would like to translate this sentence" translated into ten languages:

1. **Spanish**: Me gustaría traducir esta oración.
2. **French**: J'aimerais traduire cette phrase.
3. **German**: Ich möchte diesen Satz übersetzen.
4. **Italian**: Vorrei tradurre questa frase.
5. **Portuguese**: Eu gostaria de traduzir esta frase.
6. **Dutch**: Ik zou deze zin graag vertalen.
7. **Russian**: Я хотел бы перевести это предложение. (Ya hotel by perevesti eto predlozhenie.)
8. **Chinese (Simplified)**: 我想翻译这个句子。 (Wǒ xiǎng fānyì zhège jùzi.)
9. **Arabic**: أود أن أترجم هذه الجملة. (Awad an atarjim hathihi aljumla.)
10. **Japanese**: この文を翻訳したいです。 (Kono bun o hon'yaku shitai desu.)

Let me know if you'd like more translations!

- Roughly Before transformers (a certain type of neural network) became feasible, translation was word-for-word translation
- Transformers are designed to understand the meaning of sentences
- In 2025, translation is easy for a machine

Proof assistants have their own language



- **Proof assistant** = software to assist with the development of formal proofs
- **Example** Lean, as displayed above
- **A main issue** is to translate a human written proof into machine language

Enter, the theorem

Codex (transformer-based) assists with formalization in Lean

Codex has $\sim$ 75% accuracy when formalizing undergrad level proofs

Codex = a large language model for autoformalization

- Autoformalization = semantic parsing for the domain of mathematics
- Here an example Don't look at the bottom ☺

```
/--‘theorem‘
Power Set is Closed under Intersection
Let $S$ be a set.
Let $\powerset S$ be the power set of $S$.
Then:
:$\forall A, B \in \powerset S: A \cap B \in \powerset S$
‘proof‘
Let $A, B \in \powerset S$.
Then by the definition of power set, $A \subseteq S$ and $B \subseteq S$.
From Intersection is Subset we have that $A \cap B \subseteq A$.
It follows from Subset Relation is Transitive that $A \cap B \subseteq S$.
Thus $A \cap B \in \powerset S$ and closure is proved.
{{qed}}
-/
theorem power_set_intersection_closed {α : Type*} (S : set α) : ∀ A B ∈ 𝒫 S, (A ∩ B) ∈ 𝒫 S :=
begin
  -- $A$ and $B$ are sets. $A$ and $B$ belong to power set of $S$
  assume (A : set α) (hA : A ∈ 𝒫 S) (B : set α) (hB : B ∈ 𝒫 S),
  -- Then $A ⊆ S$ and $B ⊆ S$, by power set definition
  have h1 : (A ⊆ S) ∧ (B ⊆ S), from by {split,apply set.subset_of_mem_powerset,exact hA,apply
      set.subset_of_mem_powerset,exact hB},
  -- Then $(A ∩ B) ⊆ A$, by intersection of set is a subset
  have h2 : (A ∩ B) ⊆ A, from by apply set.inter_subset_left,
  -- Then $(A ∩ B) ⊆ S$, by subset relation is transitive
  have h3 : (A ∩ B) ⊆ S, from by {apply set.subset.trans h2 h1.left},
  -- Hence $(A ∩ B) ∈ 𝒫 S$, by power set definition
  show (A ∩ B) ∈ 𝒫 S, from by {apply set.mem_powerset h3},
end
```

Let's dream

Name	Latest version	Developer(s)	Implementation language	Higher-order logic	Dependent types	Small kernel	Proof automation	Proof by reflection	Code generation
ACL2	8.3	Matt Kaufmann and J Strother Moore	Common Lisp	No	Untyped	No	Yes	Yes[2]	Already executable
Agda	2.6.4.3[3]	Ulf Norell, Nils Anders Danielsson, and Andreas Abel (Chalmers and Gothenburg)[3]	Haskell[3]	Yes[citation needed]	Yes[4]	Yes[citation needed]	No[citation needed]	Partial[citation needed]	Already executable[citation needed]
Albatross	0.4	Helmut Brandl	OCaml	Yes	No	Yes	Yes	Unknown	Not yet Implemented
Coq	8.20.0	INRIA	OCaml	Yes	Yes	Yes	Yes	Yes	Yes
F*	repository	Microsoft Research and INRIA	F*	Yes	Yes	No	Yes	Yes[5]	Yes
HOL Light	repository	John Harrison	OCaml	Yes	No	Yes	Yes	No	No
HOL4	Kananaskis-13 (or repo)	Michael Norrish, Konrad Slind, and others	Standard ML	Yes	No	Yes	Yes	No	Yes
Idris	2 0.6.0.	Edwin Brady	Idris	Yes	Yes	Yes	Unknown	Partial	Yes
Isabelle	Isabelle2024 (May 2024)	Larry Paulson (Cambridge), Tobias Nipkow (München) and Makarius Wenzel	Standard ML, Scala	Yes	No	Yes	Yes	Yes	Yes
Lean	v4.7.0[6]	Leonardo de Moura (Microsoft Research)	C++, Lean	Yes	Yes	Yes	Yes	Yes	Yes

- Proof assistants have been around for Donkey's years
- Another use of proof assistants is in software and hardware verification
- Me dreaming Take a screenshot of a proof and it gets immediately formally verified

Thank you for your attention!

I hope that was of some help.