A Computer's Quest to Figure Out What an "Innoventor" Is

Yuval Pinter, BGU CS; uvp@cs.bgu.ac.il; www.yuvalpinter.com

Prologue: Child Meets Blend

Consider N, a six-year-old girl, walking down the street and overhearing a conversation where one person mentions a brunch date.

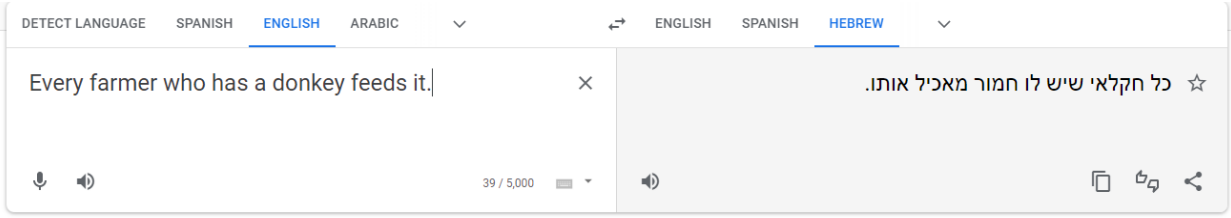
(1) - Dad, that person was talking about “Brunch”. What does that word mean?
- Well, let’s see. He mentioned types of food, do we know a similar word that is about food?
- Lunch?
- That’s right! This takes care of the *unch* part. What about *br*? Any ideas there?
- … … breakfast?
- Indeed. Now you can guess what the whole word means!

N now knows about **lexical blends**, and knows how to disassemble them and assemble the originating words using **form** (the phonemes/characters in the blend) and **context** (the conversational cues).

A Brief (?) Introduction to NLP

Natural language processing (NLP) is a field within **computer science**. As such, it asks questions that interest computational theorists, practitioners, and engineers:

(2)
- (a) How can an automatic system answer questions presented to it as free text?
- (b) How can one translate text from one language to another?
- (c) How can one interact with a human in order to fulfill a need or carry a conversation?

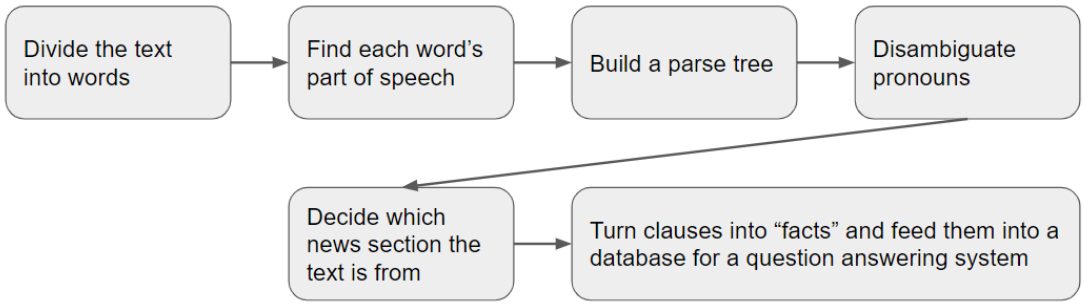


These high-level questions breed **systems** which form the “human-facing” aspects of NLP technology. But if we dive deeper, we find that many systems can use the help of more modest **tasks**:

(3)

(a) Given a sentence, can we generate a parse tree for it? (Under some formalism)
(b) Given two sentences, a “premise” and a “hypothesis”, can we infer whether the hypothesis follows from the premise?
(c) Given a paragraph, can we find out which section of the news it came from?
(d) Given a paragraph, can we find which parts (phrases) denote entities in the world?
(e) Given a pronoun in a paragraph, can we find out which entity it’s referring to?

For many years, both types of questions interested NLP researchers. System design often involved putting together separate modules that each solve a smaller task. An earlier, more basic task (**upstream**) feeds its output to a more contextualized, or more intricate task (**downstream**) until there’s enough richness in our representation of the input text to reach the final goal. For example:



Under this **pipeline** approach, it becomes important to define measures of how well a module works on a task, or **metrics**. If a new method turns out to produce better results, it can replace the old one.

(4)

(a) (cf. 3a) How many of the phrase boundaries found by the system in this set of sentences are correct?
(b) (cf. 3c) How many paragraphs from the sports section were correctly identified as such?
(c) (cf. 3e) How many pronouns were connected with the correct antecedents?

In different applications, we might care about different metrics more or less. A financial system built to generate investment advice wants to identify financial stories (and the entities within them), but can confuse sports and politics stories rather freely.

Another important point is that in order to evaluate a system against a metric, we need to know what the output of the system on the inputs should be. This question of **labeling** often involves considerable resources and differing opinions on the level of expertise required from the people who do the labeling, who are often linguists.

In the early days of NLP, tasks were approached using **rule-based** systems. A simple example would be applying a lexicon to a paragraph in order to find its topic: “if the word ‘bank’ appears, predict that it’s a *financial* paragraph”. Over the years, NLP has become a core application field for **machine learning** (ML), a different approach which focuses on **data**. This shift came mostly from an understanding that rule-based systems “don’t scale” to human language: vocabularies are too rich and dynamic; grammars are incredibly complex; variation within and across languages is difficult to generalize—and dealing with text from all sources means non-native language features heavily in the mix as well. ML takes away much of the brunt from the human developer/scientist, and tries to detect the patterns of a given task using calculations.

Concretely, to work on a task with ML we need the following:

(5)

(a) **Training data**, which is a set of examples (“instances”) built like the examples we evaluate on (which are now called **test data**). In general, the more **training** data we have, the merrier;

(b) A **model**, which takes the “input” part of the data—a sentence, a paragraph, etc.—and applies some function on it whose output is the “label”—a parse tree, a translated sentence—which can then either be evaluated or used for **updating** the model’s function. This definition is *very* abstract and can subsume a huge number of families of functions;

(c) An **objective**, which tells us for each instance how right the model was, based on the model’s prediction and the true instance label; and

(d) An **update rule**, which tells the model how the error in its current prediction can be reduced through changing tweakable parts of the model, such as different parameters in its function or the means by which the model represents the input.

The easiest kinds of models to build, ones which appear to capture updates from many different kinds of tasks well (both in NLP and in other application fields), are **feedforward networks**. They take lists of numbers as input and perform iterative calculations over them. Each of the intermediate values are also lists of numbers, and the output is as well (we later turn it into something fitting the task), and each of these lists are called **vectors**. Most of the calculations in the middle are just multiplication and addition; the numbers that the inputs are multiplied by are the ones being updated throughout the model’s training, so we call them **parameters**. The values they end up with after training can be thought of as a “fingerprint” of the specific model instance (which relies heavily on the task and the training data).

Example: we represent a sentence as an input of 5 numbers (for example, we count how many times five certain words appear in it). We use a two-layer feedforward network where the first layer transforms the input to a 2-number vector, and the second takes that intermediate vector and turns it into an output of size 3 (say: how likely this sentence is to be from politics, sports, or finance). The total number of parameters is:
First layer parameters: 5 (input) x 2 (intermediate) +
Second layer parameters: 2 (intermediate) x 3 (output) =
35 + 21 = 56.

This is a tiny model which is unlikely to learn anything. The ones we will soon get to have hundreds of millions to hundreds of billions of parameters. They will also have different types of calculations within them and different structures by which the parameters interact with each other, and this will define a network's **architecture**. These more complicated models and their intricacies accommodate for all kinds of properties that we can expect in language data:

- Word order is important;
- Language is structured hierarchically (phrases exist and compose); or
- Words can be broken into morphemes or other meaningful parts.

These accommodations can take place in the structure of the model, or in the way the input is represented, or in the way we update the model. Exploring model architecture and its effect on task performance is the mainstream of today's NLP research interest.

The Language Modeling Revolution

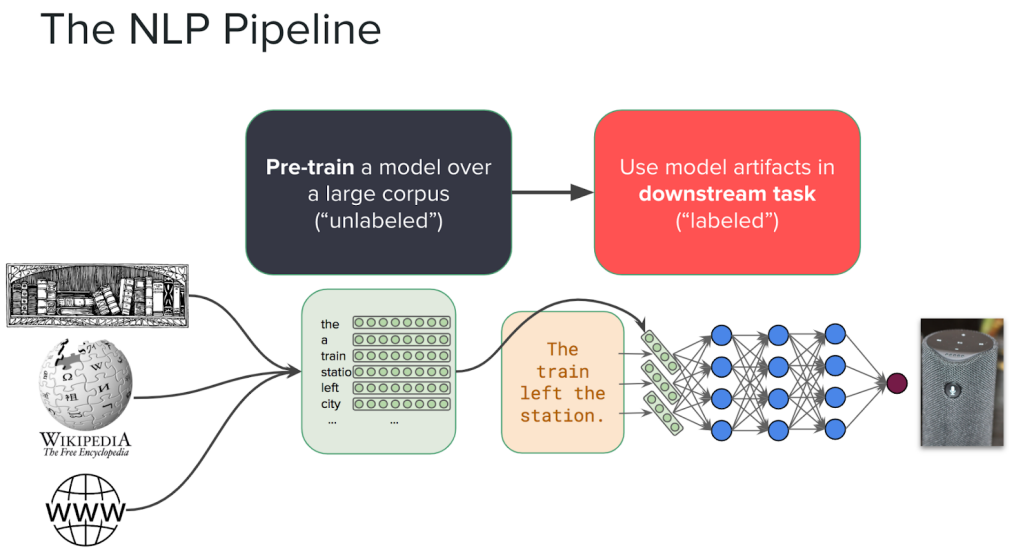
As the field was developing, the problem known as **language modeling** was mostly being treated as a simple example for class demonstrations: given the beginning of a sentence, *what is the next word*? There's very little linguistic value to such a question: it ignores structure, it's often arbitrary, it can't give us insight. Its limited usefulness came mostly from a more generalized form, the question of *what probability does this sequence of words have as text in this language?*, at which point any system whose end goal is to **generate** text, like a translator or a summarizer, can calibrate itself to this auxiliary question. With each word we generate, we can try and make sure we're not violating the grammar or talking nonsense, because ungrammatical and nonsensical utterances should have low probabilities in a language model.

One huge advantage of language modeling as an NLP task is that **it does not require manual annotation**. All one needs is a big corpus of existing text as the data. Now each prefix sequence is an input training instance, and the output is the next word, so if we have 8 billion words in Wikipedia, we can train a system on 8 billion instances, and for machine learning purposes this gives us a very well-trained model.

It turns out that language modeling is a lemon that can produce a whole lot of lemonade. Over time, people started noticing that training systems to start out as language models gives them a leg-up on the tasks that we care more about. If we use language modeling as an objective, and design parameters which represent words, once we have a trained model we can take those word representations and use them as inputs in a section-identification task, or a parsing task, or a task where we predict whether a hypothesis follows from a premise or not, and it's going to work very well. This practice is sometimes called **transfer learning**: using artifacts of one model (the language model) as an element in another.

We're going to call the basic representations of words (and sometimes other units of text) **embeddings**, because that's how they're known in NLP, but we can leave the etymology alone. What we need to know about them is that they're lists (vectors) of real numbers (decimals), they correspond to units of text, and they're a very good input format for all kinds of systems with

specific tasks. In the diagram below, they are these elongated ovals with little circles standing in for the numbers (“entries”) within.



Language models trained in 2020 and onwards can do things that seem very impressive. “Prompt” is the text given to a language model called GPT-3 with the intent that it completes it by generating a plausible continuation, word by word. “Sample response” is what it generated.

Prompt

```
Write a recipe based on these ingredients and instructions:

Frito Pie

Ingredients:
Fritos
Chili
Shredded cheddar cheese
Sweet white or red onions, diced small
Sour cream

Instructions:
```

Sample response

```
1. Preheat oven to 375 degrees F.

2. Spread Fritos in an even layer on a baking sheet.

3. Top with chili and shredded cheese.

4. Top with small onions.

5. Bake in preheated oven for 15 minutes.

6. Remove from oven and top with sour cream.

7. Enjoy!
```

Prompt

```
You: What have you been up to?
Friend: Watching old movies.
You: Did you watch anything interesting?
Friend:
```

Sample response

```
Yes, I watched The Omen and Troy.
```

Question: what happens when we're training (or evaluating) a task and we encounter a word that doesn't have a corresponding embedding? (This is called an **OOV**, or an out-of-vocabulary word)

One possible answer: augment the basic word-representation model with some way of breaking down unknown words and represent these parts of the words as if they were independent (so "station" gets its own embedding, because it's been seen often enough in the training phase, but the typo "stattion" or the station name "Atlit" get broken down into "stat"+"tion" and "At"+"lit"). These are called **subword** representations.

Language Model Meets Blend

The performance of language model-based systems on accepted metrics, and the general tendency to be impressed by the stuff they come up with, have brought forward two notions in the NLP community:

(6) Language models *understand* human language.
(7) The contextual elements of language models are so strong, that representing words doesn't matter that much anymore.

The first notion has attracted much criticism. There are philosophical arguments to be made, from the question about the role of consciousness in understanding, to wondering about what an **ungrounded** model, i.e. one that doesn't have a realization of 3-dimensional space and the shapes within it, can do with any kind of so-called "knowledge" it possesses. There's a useful analogy that's been made to parrots, and the probabilistic nature of the wiring inside the models has bred the description "stochastic parrots".

But the second notion goes by mostly unnoticed. If words are just defined by their contexts, then the meaning of an unseen word can just be inferred by the context in which we first see it. If we're using subwords, we just add those representations up and it's "good enough" for the translation we're generating, or the content type classification we need.

My work challenges this idea. While work on OOVs usually focuses on unintentional OOVs like typos, or dialect-specific language, I wanted to also include cases where our vocabularies are just being extended. Like my child's!

First Experiment

The New York Times archive allows single-word queries. An artist named Max Bittker decided to use this fact and created a Twitter bot that scrapes every story being published on the NYT website and finds the novel words, then tweets them out:

With Max's help, Cassandra Jacobs (now at the University at Buffalo) and I obtained a corpus of 2,600 words that were first published between November 2017 and March 2019, along with their contexts. Some of these real-world OOVs may surprise you:

(8) Purdue reformulated OxyContin so that the pills turned into a gooey, *unsnortable* mess when crushed.

(9) After takeoff, this *cyberpilot* senses that something is wrong.

(10) That can mean big money for the families of *kidfluencers*.

(See how Google's spell-checker doesn't like exactly these three words?)

The first experiment we ran given this data was a simple test of understanding how language works, how word formation works: given sentence (8), can a language model figure out that *unsnortable* is a new **morphological derivation**? Given (9), that *cyberpilot* is the application of a **free-form affix** to a common word? Given (10), that *kidfluencers* is a **blend**? With 18 such categories, and over 2,000 examples to learn from, the model should do rather well.

Why is this task important? First, to give more evidence for or against the hypothesis in (7); but also from a practical perspective: if we want to develop a more sophisticated system that doesn't have an OOV problem, it can include all kinds of different ways for handling them: a morphological derivation would be broken into its parts and re-composed from representations; a blend would be broken down and restored; a loanword can be looked up in a dictionary from the origin language; etc.

How well did the language model do? **Not so well**. The score it got on the classification task was around 0.3 on the F1 metric, which is hard to give an intuition for, but the important part is that it did roughly as well as a model that received no input except for character-sequence counts inside the word. No context, no semantics. All the magic of language modeling was useless on this task.

Second Experiment

Next, we looked at just the set of blends in our dataset. Look at the context a model has for *kidfluencers* in (10), or at (11):

(11) (...) the coarsening of the digital media landscape, which Mr. Trump has truly been an "*innoventor*" of (yeah, that is a term used in Silicon Valley) (...)

In both cases, and in many of the others we saw, it seemed that replacing the novel blend with the words that compose it—"kid influencers" and "innovative inventor", respectively—can be useful for a model, i.e. it brings the desired meanings to the text and if the model has a good

representation for these original words (we called them *bases*) then it would help the overall representation, and later on the system's performance.

So we designed a kind of "in-vitro" analysis for this set of blends: extract their representations from a contextualized language model called BERT, and see how close they are to the version of the sentence where the blend is replaced with the bases:

(12) That can mean big money for the families of *kidfluencers*. (=10 repeated)

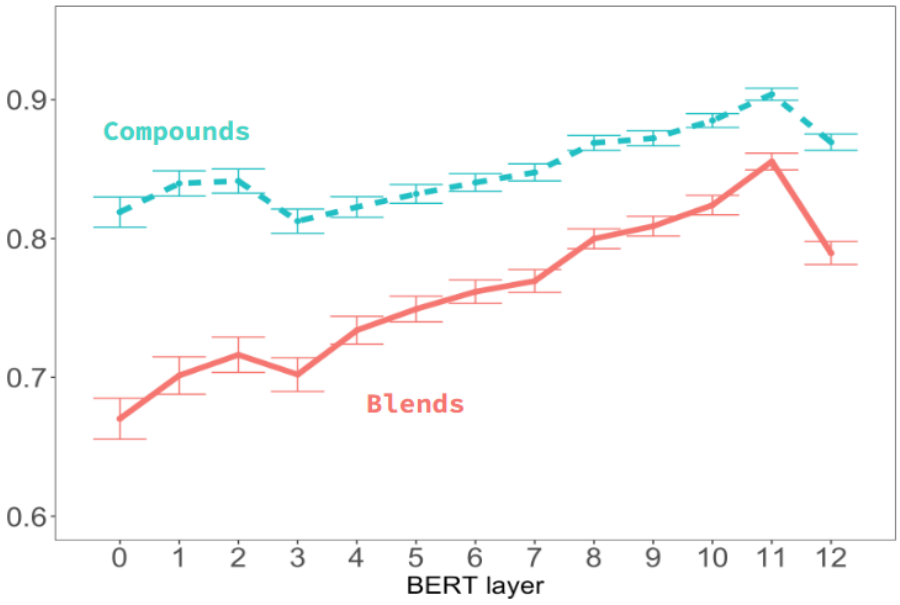
(13) That can mean big money for the families of *kid influencers*.

As a control group, we considered words that also came from our dataset but were **compounds**: two bases added together without losing characters. For example:

(14) Others write off the practice of DNA tattoos as part of the growing *bodyhacking* movement—think "neuromancer" meets "Miami Ink".

(15) Others write off the practice of DNA tattoos as part of the growing *body hacking* movement—think "neuromancer" meets "Miami Ink".

The graph below plots, for each of the classes (averaged over all items), the **similarity** between the two forms (12 vs. 13, 14 vs. 15) as a function of the model's representation layer (1.0 is the maximum). In a broad sense, layer 0 is the "form-only" representation, and as the layers progress there's more context that seeps in through interaction with the representations of other words in the sentence.



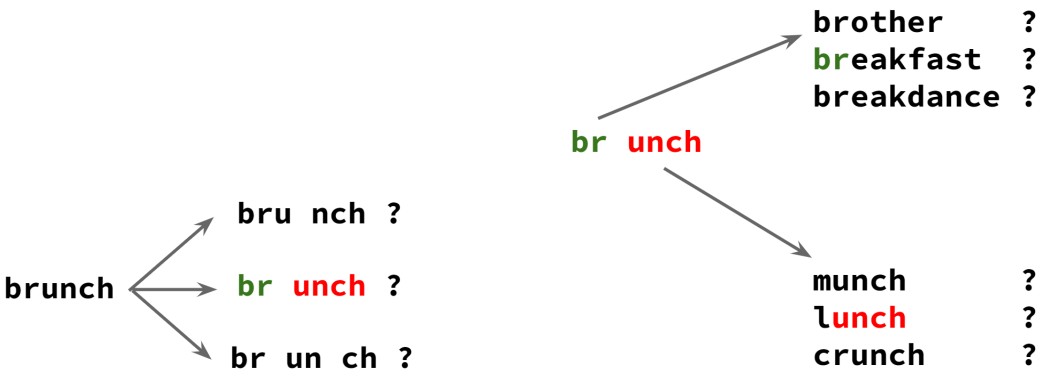
We see that even with all the context, the model doesn't represent the blends nearly as well as it could if it had knowledge of the original bases the blend came from. So if we built a system to recreate these bases, it should be able to overcome some of this challenge. But can we build such a system?

Third Experiment

We finally come back to six-year-old N. She's shown us that if she knows that blends exist, and given the form of a blend and the context she heard it in, she can figure out the meaning fairly well. We can use this insight to design an NLP task, or even two of them:

(16) Given a blend in context, produce a **segmentation** of its characters into parts that come from each base (or both, or neither). ← this task is mostly about **form**

(17) Given a context and a pre-segmented blend, **recover** the original bases from a dictionary of common words. ← this task is both about **form** and **context**



How well did the models we test on these tasks do? Again, **Not very**. The best model we tested for task (16), which relied heavily on both form and context from a language modeling backbone, managed to get the correct segmentation **25%** of the time. For task (17), the best we got was **28%** correct recovery of both bases, and an additional **39%** of only one of the two.

The ultimate test (for an NLP reader) would be to detect actual performance degradation on an end-goal task that can be traced back to problems in such novel word analysis; this is something we haven't gotten around to yet.

Epilogue: We Could Use More Linguistics in NLP

This series of experiments is one example of many that illustrate how "superficial" even the most sophisticated NLP systems are. They can perform a lot of the tasks that business-driven applications define, and they're good at aggregating a lot of signals from an input to make it seem like they've mastered certain kinds of prediction problems, but their encoding capabilities are actually pretty limited. At the end of the day, underneath all the data there's **language**, and its richness of phenomena is something that a smaller and smaller share of NLP scientists and engineers are familiar with. Model diagnostics practices, like the one demonstrated in the second experiment above, are getting more commonly used, but there's a difficulty in aligning them with structures and patterns that reflect linguistic constructs, that allow linguistic inquiry. Tasks that zoom into specific language knowledge, like the ones in the third experiment, are hard to devise and hard to follow up with development of better, more language-aware models. I've given some more examples of NLP work that follows linguistic insight in the references:

tracking a model's internal state to see how it captures subject-verb agreement; probing a model's innards to try and find entire syntactic trees; defining tasks related to claim veridicality; and there's more. Myself, I have plans and ongoing projects including attempts to improve subword language models, and to test the limits of models based only on phonological representations, and to offer a role for NLP frameworks and systems in language revitalization efforts.

But there can also be a benefit for linguists from acquaintance with the NLP landscape. While we're usually not trying to mimick human minds, we've gathered data and experience in large quantities. I would be surprised if a seasoned linguist going over a system's output won't find analogues to human behavior previously unexamined, or draw inspiration to tackle a well-known issue using insights from a computational component that performs well. I'm leaving this part open—now you know where to find me.

Selected References

[1] Yuval Pinter, Cassandra L. Jacobs, Max Bittker. 2020. NYTWIT: A Dataset of Novel Words in the New York Times. In Proceedings of the 28th International Conference on Computational Linguistics.
[2] Yuval Pinter, Cassandra L. Jacobs, Jacob Eisenstein. Will it Unblend?. In Findings of the Association for Computational Linguistics: EMNLP 2020.
[3] Alexis Ross and Ellie Pavlick. 2019. How well do NLI models capture verb veridicality?. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP).
[4] Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the Ability of LSTMs to Learn Syntax-Sensitive Dependencies. Transactions of the Association for Computational Linguistics.
[5] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP.
[6] Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On Meaning, Form, and Understanding in the Age of Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.
[7] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, Shmargaret Shmitchell. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜. In FAccT '21: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency.