Introduction to Natural Language Processing

Druhan Rajiv Shah

Spring - 2023

Contents

Course Structure

- 20% Assignment
- 40% Project
- 10% Midsem
- 10% Quizzes
- 20% Endsem

There used to be Vivas, but not anymore. (too many damn students lmao)

Objectives

- Demonstrating knowledge of the fundamental building blocks of NLP
- Apply NLP techniques for classification, representation and parsing
- Demonstrating the use of Dense vector representaton for NLP
- Explaining concepts behind distributed semantics
- Discussing approaches to global and contextual semantic representation
- Applying the above concepts for fundamental NLP tasks

Syllabus breakdown

1. Stages of NLP
2. Morphology, POS tagging, Chunking
3. Syntax
4. Distributed semantics
5. Contextual distributed semantics

Admin stuff For deadlines and all that, there will be a sort of Course Almanac shared later, based entirely on the Institute Almanac. Attendance will be managed by the Institute.

1 Introduction

Potential motivations

- Fascination with existing models like ChatGPT, Lambda
- Translation systems
- Strengthen theoretical base (?)
- Interesting applications
- To create something (AI, perhaps?)

Language Language has three main modalities (text, speech, and sign), and each of them overlaps with each other (a lot going on tbh).

This course will mainly cover the text side of things.

What's going on? Processing natural language has been the final target of Computer Science since the very beginning (ever since Ada). But Computer Science is just doing math but quickly. Math can pretty much be attributed entirely to Panini, who worked primarily on Logic. Converting Language into a Logical form, providing structure is what he's mainly known for.

Enter Linguistics, stage left.

Someone much later tried to create a much more formal structure of languages. That is now known as either Formal Language and Automata Theory, or Theory of Computation (à la Chomsky).

Theory of Computation Four levels of languages:

1. Regular Language
2. Formal/Context-free Language
3. Recursive Language
4. Recursively Enumerable Language

This leads to much more fundamental problems like the Halting problems or Satisfiability problems.

iNLP time This course is not about modern, State-of-the-Art models, but about the theoretical base that those models are built on.

This is going to be a very fast-paced course.

The focus of this course is not coding, but there will be coding-based assignments and all.

History lesson Modern computer science is considered to begin around Alan Turing's time. So pretty much most of Computer Science is based on code-breaking. Turing's code-breaker was gifted by the British Intelligence Services to the CIA and a company was born out of it. Code-breaking was pretty much all that the CIA did back then. The commpany that it collaborated with was AT&T (Bell Labs back then), which did most of the research on code-breaking which led to the development of NLP. The code-breaking work led to the development of Information Theory, which led to Coding Theory. Some of the lead singers in this concert were Claude Shannon and Huffman.

Monitoring communication is essentially what led to the development of Computer Science itself, so it makes sense that Natural Language is a major focus even now.

Language shenanigans We mainly deal with language in a sequential manner (as opposed to something like Arrival). So time(order of some sort) has to be considered regardless of language. For years, most NLP models handled text sequentially, until the advent of Transformers, which process text as a group, sort of parallelly. The sequential treatment made sense because speech is dealt with that way.

Beyond the grammar of language, there is the layer of meaning, which is a lot harder to deal with (see Dense vector representation and distributed semantics as opposed to Formal logic representation).

The '97 paper on distributed semantics marked the beginning of the modern age of Language processing, which is primarily meaning-based.

2 'Nuff Philosophy

If we disregard the socio-political factors for the development of this field, there is a certain fascination for the sheer complexity of natural language as we speak it, which we take for granted only because we use it all the time.

So what makes a language a language? Prairie dogs have developed modes of communication to convey messages of whether the thing approaching the den is a threat or not. In fact, these messages are extremely detailed. Crows and parrots are well known to be capable of developing intelligent communication.

There are things that language allows us to do in communication, which when we think about it, are extremely fascinating.

Essentially, at its core, NLP is an attempt to understand how the human brain works in dealing with language. NLP is all about computationally understanding language in order to use it for something else.

Processing of Natural Language

- Speech and text processing
- Identification of language phenomena (grammar, writing systems, etc.)
- Understanding language content (semantics, pragmatics, sentiment, etc.)

Systems Consider the OS definition of a system. A system is when multiple parts or components interact with each other to perform functions.

By this logic, language is a very complex system, consisting of several components. The typological structures that we are familiar with, can be claimed to be there for a reason. These structures incentivize us to structure our thoughts in a way that makes it efficient to convey the information necessary for communication (an organically developed protocol, so to speak). These are sort of analogous to evolution, where there are some things that just make sense, which is what ends up becoming the norm, due to selection. This is also what most likely happened with language; the documentation of the structures that we study is only a post-development analysis.

Language as we know it is not bound by any physical laws, the only mechanism that restricts language is the mode of communication, be it sound or speech. In the case of sound, the human vocal tract is only able to produce so many sounds.

Oops, more philosophy One can even say, the most chaotic system possible is human society; Asimov's works almost all verify and build on this idea. Of course, one can apply the physical definition of chaos to this system, and calculate entropy and all that, but that's another rabbit hole entirely.

Okay, back to NLP Simple keyword matching is the most basic method of dealing with things. Matching documents that contain certain keywords is something literally any layperson can do, though.

2.1 Generations of NLP

Expert systems Back in the good ol' days, there were all hand-crafted knowledge-engineering systems were all the rage (Expert systems so to speak, see MDL notes). There was significant linguistic knowledge, even computational knowledge, but all systems were knowledge-based. So pretty much rule-based systems as in CL2.

Machine Learning The next generation decided that this built up too much a priori cost and was too labour intensive. Machine Learning started with the idea of learning rules. Thus, we can use ML systems to learn those rules to set up the rule-based systems on their own. Step 4: Profit.

This led to an increase in the use of disjunctive logic as opposed to conjunctive logic.

An ML system allows us to abstract things out and can be applied in far more fields than rule-based systems, but can have possibilities of the model learning rules that are not actually valid. This requires the manual introduction of a bias in the model. The easiest way to implement such disjunctive logics with bias, is probabilities. Note that conjunctive probabilities can simply be comverted to disjunctive using complements (ggez). Sometimes these disjunctive conversions are made in order to make computation easier or even possible.

Again, pay close attention to MDL for this.

Rise and fall of Neural Networks If all this probabilistic trickery ca be simplified into a function based on some parameters, then all one needs to do is to devise that function mapping a vector of attributes onto a class space. Perceptron-based learning handles linear function approximations of pretty much everything. Marvin Minsky pointed out that this is infeasible for several reasons. This ended up killing pretty much all work in the Neural Network field.

Most of language technology, however, was developed in this era. It would be a great disservice to ourselves.

Unsupervised Semantics The most recently developed field of learning systems, are especially applicable in language systems because they are entirely based on semantics. The attempts to represent ideas using mathematical vectors was inefficient for reasons that will be covered later.

2010 saw the development of Word2Vec technologies (as opposed to word embeddings), which blew this field up suddenly.

2.1.1 Learning styles

Types of ML models There's either Supervised, Unsupervised, or Self-Supervised learning. Supervised is based on manual approval or disapproval of classifications. This could be done by pre-annotated data as well.

Note that pretty much every problem can be boiled down to some sort of Classification problem.

Self-supervision? You mean slacking off? Not quite a misnomer. It's all about converting unsupervised data into supervision, sort of. Considering one plain sentence with no expert supervision, then we can't get a lot from it on its own. But, by replacing a word with a class and we can obtain some pseudo-classification problem-like situation. This can be done only if the model 'understands' what the surrounding words mean. Here, the meaning of 'understand' kinda gets messed up, but don't worry about it.

Now, given some more data, the model can actually compute a distribution for the possible replacements for the removed word (This is called a Mask prediction model).

2.2 Structure of language

- Words
- Syntax
- Semantics
- Discourse
- **Bonus:** applications exploiting each of the above

Despite dictionary propaganda, words themselves don't have meaning, they obtain their meaning from context. Similarly, each of these fields relies deeply on each other.

A Challenger approaches Ambiguity can be context-induced or structural (despite dictionary propaganda), the latter of which can be context-interpreted. PP-attachment ambiguity is an example of structural ambiguity. Some such ambiguities can't even easily be resolved by humans either! Headlines are even worse (Foot Heads Arms Body :P).

Language allows for a lot of creativity, which is what computers pretty much hate. After all, isn't creativity just a resolution of entropy, which would lead to this being a chaotic system?

Internet shenanigans The advent of the internet boom, especially considering the incrased reach off late, has posed several challenges in language processing. One such challenge is language evolution which is caused due to this increased reach. This also leads to another challenge which is scale. There is simply way too much data on the internet to reasonably handle as of now.

2.3 Finally, Natural Language Processing

This course We will be progressing from Morphology and Syntax all the way upto touching a bit of Discourse analysis. In this course, we will be focusing on text-based NLP, ignoring speech and sign.

3 Morphology and Syntax

Tokenization Recall CL2. Consider the sentence "This is my brother's cat." Considering "brother's" as its own token is dangerous because the factor of 'brother' is completely lost in it. However, not all apostrophe-ending words are to be considered similar.

Punctuated with some Writing Space-based tokenization is infeasible because there are several languages (Traditional Mandarin, Modern Thai) that don't even use spaces. (Ni-ho-n-go-ga-wa-ka-ri-ma-su-ka?) Some languages use spaces between adpositions, while some don't (Hindi vs Gujarati lmao). The existence of Sandhi or rendaku (compounding, even) alone is sufficient example for how spacing is not absolutely necessary. Language on its own does not use punctuation. There are extra-syntactic features, but there is no strict punctuation as is. In fact, a person can use any pause length or frequency as they wish.

Old Sanksrit poetry uses only two punctuation marks because of the oral tradition that led to the advent of Sanskrit as a language. In fact, writing as a system are a very recent phenomenon as compared to language as a whole. One only needs punctuation to indicate pauses. Essentially, punctuation is just a printer's device.

Terminology

Type is a word which might be present in its root form in the Dictionary

The set of all types in a language is its vocabulary

A high type-token ratio indicates that a language has a rich, variegated morphology. This could even imply agglutinativity. Marathi has 97 freaking morphological forms for verbs lmao.

Is tokenization difficult? When it comes to tokenization, types are no longer of any concern to us. One can use rule-based tokenization, but that's too messy. One can let the machine itself learn how tokens are split, which has its own challenges. Hyphenation, abbreviations, numbers, dates and punctuation can pose problems on their own.

Regular expressions are rampant here :P

3.1 Language Models

Models are a set of parameters that numerically describe reality. So it's essentially just a bunch of numbers.

Now, models are not necessarily perfect. Rather, they aim to describe reality as opposed to strictly being perfect. The process of modelling is constantly refining those models.

This automatically destroys the 'is AI sentient' argument :P

Good enough models can attempt at omniscience, but it's just a farce.

3.1.1 N-grams

What can we do with raw data? From raw data, can we learn a set of parameters that performs a certain task? Claude Shannon proposed that given a sentence, can we how likely it is to belong to a certain language? Now, considering sentence length, we don't use long sentences very often. This would mean that the longer the sentence, the less likely it is to belong to a language (Enter Zipf). Simply considering probabilities of groups of words or sentences is sorta obviously messy.

Let's not Mark-this-ov Here, we make a 'Markov assumption': A word depends on the 'prior local context', which is the previous few words. The length of the 'prior local context window', say n, would make this an n-gram model (3-length window would make this a trigram and so on).

From a linguistic perspective, we already know that there is a structure. However, computationally, we only see strings of words. And in this context, the words can only depend on their surroundings.

By looking at bad enough examples, we can see that given long enough sequences, the data can get very unreliable. Small counts are usually very unreliable, because given large enough n, we can easily get absurd probabilities.

However, there is another gain from this. Were language truly random, probabilities would give absolute answers regardless of n, because true randomness could lead to any possible sentence, equally frequently. This is an argument against the chaotic nature of language.

Also, computationally, given large enough data, the number of bins in the n-gram models grows exceptionally large. We also need to include combinations that we haven't seen in the data before, but are possible in the language in order to model it effectively.

Mar-kons

- The sparseness of words and word clusters can lead to very spread out distributions, which can lead to messy and inaccurate predictions.
- Dealing with unknown words in the prediction output can lead to straight up zero probabilities, which is messed up, yo.

3.1.2 Evaluation and Data Sparsity

Rather perplexing, innit? Perplexity and Entropy are usually the measures used to evaluate how good a model is. Specifically, it measures how well your model fits a corpus, once it's built.

Both of these are based on Information Theory, which deals with chaotic systems and measuring the degree of chaos in a system. Entropy is a measure (heuristically) of the extent of chaos in a system, while perplexity is the measure of how much confusion a model can have when trying to determine the outcome of a system. (See how vague this is?)

Specifically, Entropy is the lower bound on the number of bits needed to encode information (say, about the n-gram likelihood). Perplexity is the average number of choices that can be made, weighted by their probabilities.

$$\text{Entropy } H(W) =$$
$$\text{Perplexity } PP(W) = 2^{H(W)}$$

Perplexity being a measure of the number of choices to be made, can be used as a metric for determining how well a model approximates something. For example, when matching a model to someone's language style, the fewer the choices required to determine the next possible word in a sentence, the closer the match, so a lower perplexity would indicate a better match.

3.1.3 Zipf it!

When the population of several physical, natural and social systems (like words in a language, or species in a ecosystem) is plotted with their frequencies, the resulting plot is a reciprocal-like curve (Zipf distribution). Thus, when plotted on a log-log graph, the resulting curve is a straight line.

Another way of putting this, is that when ranked (1-indexed), the product of the frequency of a word and its rank is approximately a constant.

Note: Do not confuse this with Dirichlet's Theorem which considers the residue classes of primes to be evenly distributed.

This makes sense for languages, because naturally, function words would be way more frequent than, say, proper nouns. Anaphora and Specifiers make up some of the most frequent words in English at least.

At this point, bringing up the issue of deciding whether to include stopwords in models is necessary. Sometimes, when the functions played by stopwords are not necessary for our model, it makes sense to exclude them, but when considering n-gram models, it makes sense that stopwords be included since they would affect the sentence formed due to their large probabilities.

3.1.4 Back to N-grams

Now, when considering sparse distributions or while encountering new words to predict, then the model can either mess up or fail entirely. This needs to be countered.

1. We can reduce the value of n, possibly down to 1, but that cannot overcome the issue of new words.
2. We can define probabilities as $\frac{f_w+1}{\sum_w (f_w+1)}$ (Again, very ad-hoc)
3. We can group words into categories, each with their own probabilities, and then uniformly distribute the lexicon of that category.

4. We can assign a function of n-gram probabilities of the context and all possible follow-ups to the probability of the context with a new word. This covers the issue of new words, but doesn't deal with unseen contexts instead of new words.

5. We can use idea 2, but with $\epsilon > 0$ instead of 1.

Aight buddy, back it up now Idea 1 combined with Idea 4, implemented with considering a certain function of the counts without explicitly considering the empirical probability definition, can actually work out. Such a model is called a Back-off model. This is because every time a new word is obtained in the context of a word, we back off down to a smaller n value, and for new words to be predicted, we follow Idea 4.

In a way, we are also using the Idea 5 in the sense that we aren't using the conventional definition of empirical probabiliy which uses purely word counts.

Damn that's smooth Granted, these are computationally intensive, because we would need to compute multiple large distributions, one for each n, instead of simply one large distribution. Now, this can be simplified using tricky computational methods. (To be updated) Potentially, interpolation can be used to simplify this.

Such improvements in models, including interpolation, are called smoothing techniques. See:

- Laplace smoothing
- Lidstone's Law
- Held-out estimation
- Good-Turing estimation

3.1.5 Missing class

Lorem ipsum

3.1.6 Holding out on estimates

$$lorem$$

Where N_r is the number of n-grams with frequency r in the training data and $T_r = \sum_{i \in [N_r]} f_{h_0} (= r \cdot N_r)$. Now, we also define the reestimated frequency $r^* = \frac{T_r}{N_r}$

Thus, the probability is calculated keeping both training and held-out data in mind. But which of these works better? What happens if the two of them are switched?

This can work if we consider both and simply average them out. One way is $\frac{r^*_{01} + r^*_{10}}{2}$. But this is a little too weird. Jelinek and Mercer (1985) recommended

$$\frac{T_r^{01} + T_r^{10}}{N(N_r^{01} + N_r^{10})}$$

3.1.7 Turing's kinda good no?

It is natural for data for some frequency bins to be rather empty all of a sudden. Now, what if we consider words that appear very rarely and treat them as we would treat other equally rare words? Isn't a word that appears only once pretty much the same as one that doesn't appear at all?

Once there is more structure to the data, this estimation becomes less and less valid. However the claim is that this becomes valid for larger and more natural data, so to speak.

Additionally, if we assign probabilites in the classical way, elements with frequency 0 will always have a 0 probability. This defeats the entire purpose of smoothing, which is to deal with such words.

Thus, we can recalculate adjusted probabilities for a rarer word as

$$r^* = (r+1)\frac{E(N_{r+1})}{E(N_r)}$$

And then we consider the adjusted probability as $\frac{r^*}{N}$

Assignment 0.5

Read the paper Simple Good-Turing Algorithm and prepare an around-1-page report.

3.1.8 Why so smooth?

There are always some sentences, however, that a human would find very natural, that a model would not even consider. This is because the model only considers immediate surroundings and that too linearly. That is not how a human would read it (Recall ITL1 and CL1).

Yeah that's right, back it up Back-off estimation is essentially changing the value of n to deal with unseen words as mentioned earlier (See paragraph **Aight buddy, back it up now**). It can also possibly be increasing the value of n to deal with shenanigans.

Katz (2019) We can redefine the Katz back-off probabilities as

$$P_{li}(w_n|w_1, w_2, \ldots, w_{n-1}) = \begin{cases} \alpha \cdot \frac{c(w_1,\ldots,w_n)}{c(w_1,\ldots,w_{n-1})} & \text{Counts are greater than some } k \\ (1-\alpha) \cdot \frac{c(w_1,\ldots,w_n)}{c(w_1,\ldots,w_{n-1})} & \text{Otherwise} \end{cases}$$

Linear Interpolation This ends up working really well.

$$P(w_n|w_1, \ldots, w_{n-1}) =$$

This is Witten weally Bell Witten-Bell smoothing manages to capture the robustness of interpolation while also handling the intuitiveness of back-off models. It also considers the situations where for a large number of possibilities, the history is either very 'productive' or the new word itself is prone to many possible histories.