ON THE NATURALNESS OF SOFTWARE

-- Dr. Abram Hindle et al.

Felix Wang,
David R. Cheriton School of Computer Science

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

PROBLEM TO BE SOLVED

Conjecture

Most software is also natural (like other natural languages).

- **What does natural mean?**

Attendant constraints and limitations of human work.

⇒

- Repetitive
- Predictable

⇒

- Code can be modelled by statistical language models.
- Such models are useful.

Problem to be solved

- Code can be modelled by statistical language models.
- Such models are useful.

N-gram: A statistical Language Model

⇓ Model

Code (in Java, C, Python)

⇓ Build

Eclipse Plug-in

NEW IDEA

Repetitiveness & Predictability

- Why are language models useful for this scenario?

Because Repetitiveness & Predictability appear on:

1. Lexical level -> Token sequences
2. Syntactic level -> Grammatical structures
3. Semantic level -> Probabilistic

$\boxed{\text{Statistical Language Model}}$

N-gram

- Example paragraph:

I like pink dresses. I like pink. I do not like dresses. I like green.

3-gram example

- Tokenize:

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s>: start symbol of a sentence.
</s>: end symbol of a sentence.

3-gram example

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s>: start symbol of a sentence.

</s>: end symbol of a sentence

Count all possible trigrams:

<s> <s> I: 4

<s> I like: 3

I like pink: 2

I like green: 1

I do not: 1

……

3-gram example

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s> <s> I: 4

<s> I like: 3

I like pink: 2

I like green: 1

I do not: 1

……

Predicting the next word of a sentence:

I like …

P(pink|I,like) = 2/3

P(green|I,like) = 1/3

So next word is more likely to be pink.

3-gram example

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s> <s> I: 4

<s> I like: 3

I like pink: 2

I like green: 1

I do not: 1

……

- Sparsity:

I like dresses.

It's a perfectly valid sentence, but:

P(dresses|I,like) = 0

Since there was not enough learning data.

N-gram

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s> <s> I: 4

<s> I like: 3

I like pink: 2

I like green: 1

I do not: 1

……

Solution:

1. Fall back to (N-1)-gram if the probability of N-gram is 0.
2. Kneser-Ney Smoothing:

$$P_{KN}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c_{KN}(w_{i-n+1}^{i}) - d, 0)}{c_{KN}(w_{i-n+1}^{i-1})} + \lambda(w_{i-n+1}^{i-1}) P_{KN}(w_i|w_{i-n+2}^{i-1})$$

An easy smoothing example (Laplace):

$$p = \frac{numerator + 1}{denominator + size\ of\ vocabulary}$$

N-gram

<s> <s> I like pink dresses </s> </s>

<s> <s> I like pink </s> </s>

<s> <s> I do not like dresses </s> </s>

<s> <s> I like green </s> </s>

<s> <s> I: 4

<s> I like: 3

I like pink: 2

I like green: 1

I do not: 1

……

Metrics:

Cross-entropy Loss/Log-transformed perplexity:

$$H_{\mathcal{M}}(s) = -\frac{1}{n}\sum_{1}^{n} \log \, p_{\mathcal{M}}(a_i \mid a_1 \ldots a_{i-1})$$

Rationale: How “confident” the language model is able to guess the next word.

Experiment

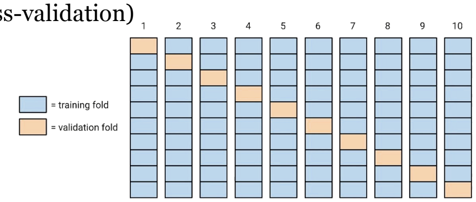
- Data:

Natural languages: Two famous corpora (Brown corpus, Gutenberg corpus).

Coding languages: Java projects, Ubuntu applications in C, after removing comments.

- Training technique (10-fold cross-validation)

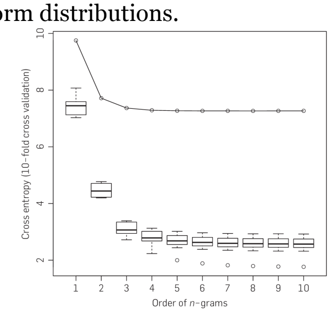
Randomly select 90% for training and 10% for validating.



RQ1:Do n-gram models capture regularities in software?

- Data: Java projects
- Findings:

1. Software unigram entropy is much lower than uniform distributions.
2. Cross-entropy declines rapidly with n-gram order.



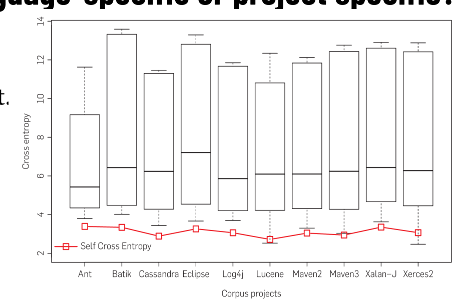
- Claim:

1. Java contains a lot of local repetitiveness.
2. Software is far more regular than English.

RQ2:Is the local regularity language-specific or project specific?

- Data: 10 different Java projects
- Train a trigram model on each project.
- Findings:

1. Self cross entropy is lower.
2. Cross-project entropy is higher.



- Claim:

Each project has its own type of local, non-Java-specific regularity.

RQ3: Similarities within app domain, and differences between domains?

- Data: 10 Ubuntu application domains
- Findings:

1. Cross-domain entropy is higher.
2. Within-domain entropy is lower.

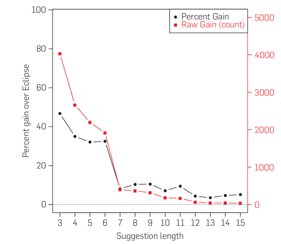
			Tokens	
Ubuntu Domain	**Version**	**Lines**	**Total**	**Unique**
Admin	10.10	9092325	41208531	1140555
Doc	10.10	87192	362501	15373
Graphics	10.10	1422514	7453031	188792
Interpreters	10.10	1416361	6388351	201538
Mail	10.10	1049136	4408776	137324
Net	10.10	5012473	20666917	541896
Sound	10.10	1698584	29310969	436377
Tex	10.10	1405674	14342943	375845
Text	10.10	1325700	6291804	155177
Web	10.10	1743376	11361332	216474

- Claim:

A lot of local regularity is repeated within application domains, and much less so across domains.

Eclipse Suggestion Plug-in

- Eclipse's built-in suggestion engine (ECSE) vs N-gram model suggestion engine (NGSE)



- NGSE: good at short tokens.
- ECSE: good at long tokens.

- MSE:

If ECSE has long tokens within the top n, take ECSE's top n offers.

If not, half NGSE, half ECSE.

Eclipse Plug-in's achievement & findings

1. NGSE works best with shorter tokens (probably because coders choose shorter names for frequently used entities).

2. MSE saved more keystrokes compared to ECSE.

	Top 2	**Top 6**	**Top 10**
$\mathcal{ECSE}$	42743	77245	95318
$\mathcal{MSE}$	68798	103100	120750
Increase	61%	33%	27%

POSITIVES

Positives #1: Deterministic & Explainable

- The choice of the N-gram model is good.
- The N-gram model is deterministic and explainable. But many language models based on optimization algorithms are not!
- The scope of this paper is to prove: Software is repetitive & predictable.
- Good for a study to "prove" a hypothesis.

Positives #2: Strong Empirical Evidence

- Large, real-world dataset
  - 10 major Java projects
  - 10 Ubuntu domains of C applications.
- Large-scale natural language corpus
  - Brown: ~1 million words
  - Gutenberg: ~2.5 million words
- Rigorous procedure: 10-fold cross-validation.
- Robust and strong empirical evidence.

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

NEGATIVES

My critiques of this paper could be fundamentally wrong due to my shallow research experience, please DO correct me during the discussion section.

Negative #1: Is repeating a good sign?

- Repetition does mean regularity, but it is not always a good sign in Software Engineering.
- Any preprocessing on the codebase that the n-gram model is trained on?
  - removed comments
  - tokenized codes
- Any legacy code? Poor practices that caused redundancy (that caused repetition)?
- Code suggestions to the programmer
  – As good as possible? Or as repetitive as possible? As a statistical model's natural rationale is to suggest something as repetitive as possible.

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Negative #2: The MSE algorithm is too superficial

- Eclipse's built-in suggestion engine (ECSE): "are typically based on type information available in context".
  ->Miss statistical patterns

- N-gram model suggestion engine (NGSE): a statistical model that considers partial semantic, lexical, and syntactic information.
  ->Miss type information.

- Can we build something by combining the useful information from 2, not by simply choosing one or another?

Negative #3: Limitations of N-gram

- N-gram's ability to capture grammatical structure is extremely limited.

- Able to: catch local grammatical structures like: ``public static void``, ``System.out.println(``, ``for (i = 0; i < n; i++) {``

- Unable to catch: ``try {//~100 words } catch {}``.
  - It will need a 100-gram model! Which will be extremely sparse
  - Obviously, there's some grammatical structure between ``try{`` and ``} catch{``.

- N-grams cannot catch some long grammatical structures.

Negative #4: The scope of this paper

- Most software is natural -> It is also likely to be repetitive and predictable

- Most software is repetitive and predictable -> Most software is natural?

- Not a sufficient and necessary condition!

> **We begin with the conjecture that *most software is also natural*, in the sense that it is created by humans at work, with all the attendant constraints and limitations—and thus, like natural language, it is also likely to be repetitive and predictable. We then proceed to ask whether (a)**

Negative #4: The scope of this paper

- The author did not formally state that "software is natural".
  It instead said: "(Software) it's regular", " (Software) it's not random", or this experiment supports the conjecture.

- Because this evaluation cannot prove the conjecture, and software really isn't fully a natural language by any means (tolerance of errors? one keyword can have multiple meanings?).

- Do we really need to prove this catchy conjecture?
  - Proving that software is repetitive and predictable is already a huge contribution!

FUTURE WORK

Context Free Grammar (CFG)

<table>
<tr><th>Grammar</th><th>Lexicon</th></tr>
<tr><td>S → NP VP</td><td>Det → that | this | the | a</td></tr>
<tr><td>S → Aux NP VP</td><td>Noun → book | flight | meal | money</td></tr>
<tr><td>S → VP</td><td>Verb → book | include | prefer</td></tr>
<tr><td>NP → Pronoun</td><td>Pronoun → I | she | me</td></tr>
<tr><td>NP → Proper-Noun</td><td>Proper-Noun → Houston | United</td></tr>
<tr><td>NP → Det Nominal</td><td>Aux → does</td></tr>
<tr><td>Nominal → Noun</td><td>Preposition → from | to | on | near | through</td></tr>
<tr><td>Nominal → Nominal Noun</td><td></td></tr>
<tr><td>Nominal → Nominal PP</td><td></td></tr>
<tr><td>VP → Verb</td><td></td></tr>
<tr><td>VP → Verb NP</td><td></td></tr>
<tr><td>VP → Verb NP PP</td><td></td></tr>
<tr><td>VP → Verb PP</td><td></td></tr>
<tr><td>VP → VP PP</td><td></td></tr>
<tr><td>PP → Preposition NP</td><td></td></tr>
</table>

Figure 18.8 The $\mathscr{L}_1$ miniature English grammar and lexicon.

- CFG: A concept related to traditional NLP aiming to catch the grammatical structure of natural languages.

NP: Noun Phrase
VP: Verb Phrase
PP: Prepositional phrases
Det: Determiner
Aux: Auxiliary

- Construct an abstract syntax tree of what grammar is allowed in the training data, along with the training process.

References: Jurafsky, Daniel, and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd edn, 2025.

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Probabilistic Context Free Grammar (PCFG)

<table>
<tr><td>S → NP VP</td><td>[.80]</td><td>Det → that [.05] | the [.80] | a [.15]</td><td></td></tr>
<tr><td>S → Aux NP VP</td><td>[.15]</td><td>Noun → book</td><td>[.10]</td></tr>
<tr><td>S → VP</td><td>[.05]</td><td>Noun → flights</td><td>[.50]</td></tr>
<tr><td>NP → Det Nom</td><td>[.20]</td><td>Noun → meal</td><td>[.40]</td></tr>
<tr><td>NP → Proper-Noun</td><td>[.35]</td><td>Verb → book</td><td>[.30]</td></tr>
<tr><td>NP → Nom</td><td>[.05]</td><td>Verb → include</td><td>[.30]</td></tr>
<tr><td>NP → Pronoun</td><td>[.40]</td><td>Verb → want</td><td>[.40]</td></tr>
<tr><td>Nom → Noun</td><td>[.75]</td><td>Aux → can</td><td>[.40]</td></tr>
<tr><td>Nom → Noun Nom</td><td>[.20]</td><td>Aux → does</td><td>[.30]</td></tr>
<tr><td>Nom → Proper-Noun Nom</td><td>[.05]</td><td>Aux → do</td><td>[.30]</td></tr>
<tr><td>VP → Verb</td><td>[.55]</td><td>Proper-Noun → TWA</td><td>[.40]</td></tr>
<tr><td>VP → Verb NP</td><td>[.40]</td><td>Proper-Noun → Denver</td><td>[.40]</td></tr>
<tr><td>VP → Verb NP NP</td><td>[.05]</td><td>Pronoun → you [.40] | I [.60]</td><td></td></tr>
</table>

References: Johnson, David, and May Young. “Course:CPSC522/PCFG - UBC Wiki.” *Wiki.ubc.ca*, 2017, wiki.ubc.ca/Course:CPSC522/PCFG. Accessed 20 Sept. 2025.

- Construct an abstract syntax tree of what grammar is allowed in the training data, along with the training process.
- Construct an abstract syntax tree, but this time with a probability.

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Combine PCFG and N-gram

- The N-gram model can work with PCFG, and it’s a well-known technique in the field of traditional NLP [1].
- Incorporate these two together, and let the N-gram model be capable of catching complicated grammar in code, and repeat the procedure of this paper again, to see if the model catches something new.
- References
[1] Pauls, Adam, and Dan Klein. "Large-scale syntactic language modeling with treelets." *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2012.

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

RATING

3.5/5 (GOOD PAPER BUT NOT LIFE-CHANGING)

The idea of naturalness of this paper is novel, the work done is rigorous, but its reliance on the N-gram model limits its ability to catch something more in-depth.

DISCUSSION POINTS

1. One and a half years after this paper was posted on Communications of the ACM, a paper called "Attention is all you need" came out, and everything about language models, computer vision, speech recognition, and machine translation changed fundamentally. For the future directions section envisioned by Dr. Hindle (section 6), which direction do you think becomes irrelevant, and which direction do you think becomes more relevant and realistic?

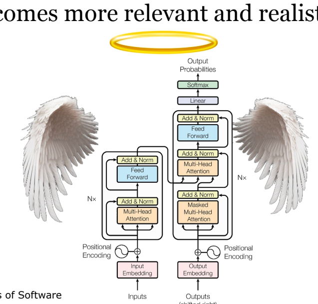


Figure 1: The Transformer - model architecture.

References: Vaswani, Ashish, et al. "Attention is all you need." *Advances in neural information processing systems* 30 (2017).

2. The N-gram-based Eclipse plug-in showed 27–61% improvement in code suggestion. And the measure used is the number of keystroke savings. Do you think this result convincingly demonstrated practical benefits, or is it more of a proof-of-concept?

What makes you think it indicates practical benefits? Or what follow-up study could be done to make it more convincing?

3. Should “Quality of Code” play into our proof for naturalness? If yes, how can we incorporate it? If no, please justify it.

4. If we now have Eclipse’s built-in suggestion engine and N-gram model suggestion engine (NGSE) both in hand, how could we build something potentially better than MSE?

5. Any disagreement or agreement with my critiques before? Do you have other positives or negatives that you want to share?