Gramble:
A Tabular Programming Language for Collaborative Linguistic Modeling

Patrick Littell, Darlene Stewart, Aidan Pine, Roland Kuhn
Digital Technologies Research Centre
National Research Council Canada

Fineen Davis
Congress of Aboriginal Peoples

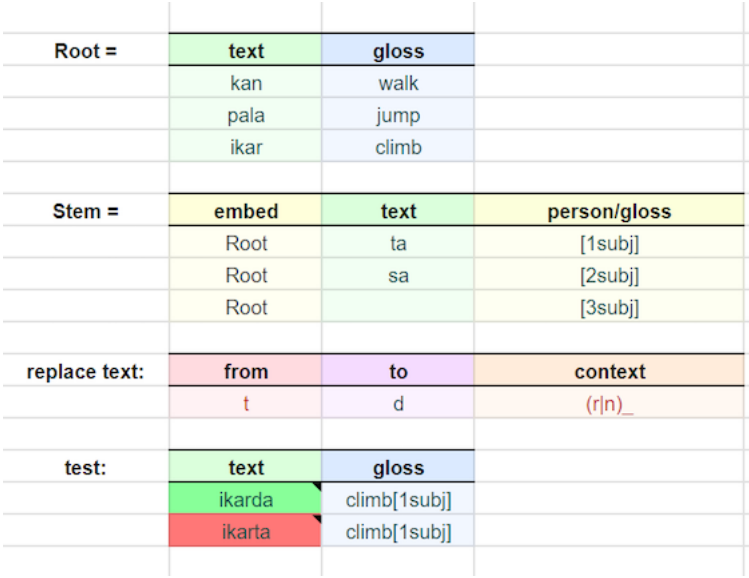
What is Gramble?

It's a DSL for collaborative grammar development

- It's meant for the same sort of handwritten language processing/generation applications as TWOLC (Koskenniemi, 1983, 1986) and XFST/LEXC (Beesley and Karttunen, 2003)
    - Morphological parsers and lemmatizers, verb conjugators, etc.
- Available under the MIT license at ***github.com/nrc-cnrc/gramble***

Designed to be much easier for non- and beginner programmers to read/write/maintain.

- Easier to incorporate subject-matter experts into development, and even inherit the project if/when the original programmer leaves.
- Uses a concise tabular syntax that our SMEs find intuitive.
- Can embed into Google Sheets for live group-programming.



Root =	text	gloss
	kan	walk
	pala	jump
	ikar	climb

Stem =	embed	text	person/gloss
	Root	ta	[1subj]
	Root	sa	[2subj]
	Root		[3subj]

replace text:	from	to	context
	t	d	(r|n)_

test:	text	gloss
	ikarda	climb[1subj]
	ikarta	climb[1subj]

What is Gramble?

Underneath, based on a novel generalization of regex derivatives (Brzozowski 1965, Antimirov 1995) to *n*-tape automata (for n > 2).

- That is, not just one input tape to one output tape, but any number of tapes to any number of tapes, in any direction.
- This might sound like it'd be more difficult to program, but it ends up being easier, because it eliminates awkward workarounds.
- Many phenomena naturally need additional "fields" or "tiers" of information. Requiring that the programmer figure out how to interleave these onto two tapes is a source of ballooning complexity as FST projects evolve beyond proofs-of-concept.

$D_c^t\varepsilon$	=	Ø	$\delta^t\varepsilon$	=	ε
$D_c^t\emptyset$	=	Ø	$\delta^t\emptyset$	=	Ø
D_c^t(x:a)	=	ε, if t = x and c = a Ø, otherwise	δ^t(x:a)	=	Ø, if t = x x:a, otherwise
D_c^t(A·B)	=	D_c^tA · B ∪ δ^t A · D_c^tB	δ^t(A·B)	=	δ^tA · δ^tB
D_c^t(A∪B)	=	D_c^tA ∪ D_c^tB	δ^t(A∪B)	=	δ^tA ∪ δ^tB
D_c^tA*	=	(δ^t A)* · D_c^tA · A*	δ^tA*	=	(δ^tA)*

The dark matter of NLP

Academic papers usually talk about low-resource NLP/speech technology in the future tense

- “Someday techniques like ours may lead to NLP for all 7000 languages of the world.”

But there’s already a lot out there, especially if you search outside the academic literature.

- Spellcheckers, verb conjugators, orthography converters, smart dictionary search, autocomplete, etc. in more languages than you’d think (Arppe et al. 2016, Littell et al. 2018).
- Much of it is never associated with an academic publication, and thus flies under the academic radar (cf. Hanselman’s “dark matter of programming”).
- Given the lack of digitized text/speech available, much of it is still based on handwritten declarative programs.
    - e.g. a handwritten FST, or custom Python scripts, or ad-hoc “little languages” (Bentley 1986).

Non-ML NLP still has benefits

Especially in educational applications

- In the absence of relevant training data, neural NLG is prone to fabrication.
- A beginner student won't necessarily be able to catch the lie!

Teachers feel they should remain the final arbiters of the answers.

- It's difficult to guarantee this when decisions about answers are made inside a black-box model.
- When neural NLG outputs incorrect or inappropriate results, it is not straightforward to fix.

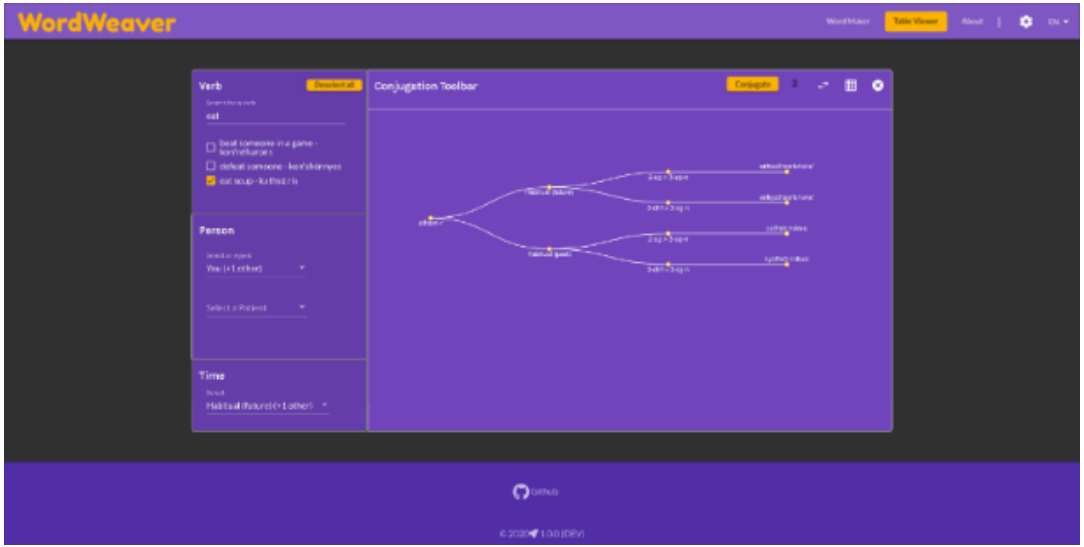
Even if neural systems worked well enough here, there's a language sovereignty issue.

- Our goal is that communities remain the ultimate decision-makers about their language.
- Delivering systems that they can't practically understand or change moves us *further* from this goal, not closer.

But rule-based NLP has its own problems

We encountered stumbling blocks as our own products evolved from proofs-of-concept to substantial, public-facing projects.

- Our team at NRC largely serves Indigenous language organizations (schools, publishers, etc.), with very limited data and human resources.
- Most of our products still incorporate human-written rules/databases at some level.



Kazantseva et al., 2018

But rule-based NLP has its own problems

As the products evolved, the code tended to balloon in complexity to the point that only the original programmer could understand and change it.

- Clients expressed concern about their inability to make heads or tails of the complex source. “What happens if/when the original programmer moves on?”
- These are educational products, after all, and curricula change over time.

This is also a language sovereignty issue!

- Does accepting a project like this commit the community to a long-term dependency on a third party?

```
define GRAMMAR GEN .O. ~$[ ə ] .O.  ~[[$ ə ]^>1] .O.  ~[[$ ə ]^>2] .O.  ~[[$ ə ]^>3] .O.
define GRAMMAR GRAMMAR .O. ~$[ ə "]" WB ] ;
define GRAMMAR GRAMMAR .O. ~$[  R "]" "N[" ə  ] .O.  ~[[$  R "]" "N[" ə  ]^>1] .O.  ~[[$
push GRAMMAR
```

```
LEXICON TensePrefix

   !!! Affirmative Tenses (with mono ku-)
   [PRES][PROG]:@U.POL.POS@@U.KU.YES@na^                ObjectPrefix;
   [PAST]:@U.POL.POS@@U.KU.YES@li^                      ObjectPrefix;
   [PAST][PERF]:@U.POL.POS@@U.KU.YES@me^                ObjectPrefix;
   [FUT]:@U.POL.POS@@U.KU.YES@ta^                       ObjectPrefix;
   [FUT]:@U.POL.POS@@U.KU.YES@ta^ka^                    ObjectPrefix;
   [PRES][COND]:@U.POL.POS@@U.KU.YES@nge^               ObjectPrefix;
   [PAST][COND]:@U.POL.POS@@U.KU.YES@ngali^             ObjectPrefix;

   !!! Affirmative Tenses (without mono ku-)
   [PRES]:@U.POL.POS@@U.KU.NO@a^                        ObjectPrefix;
```

Especially when describing morphologically complex languages, XFST/LEXC code can become very hard to read.

Investigating the pain points

But these clients had been closely collaborating with the programmer for years.

- It's a representation of their own knowledge.

At what point do they *cease* to understand the product?

- One issue was just the regex-like syntactic density of XFST.
- But also, the SMEs had been participating largely by sending the programmers tables (paradigms, etc.), which the programmer manually translates, or automatically transpiles, to executable code.
- That transformation is the main place where things go wrong – once it happens, the SMEs are no longer directly working on the system, and the transformation is irreversible.

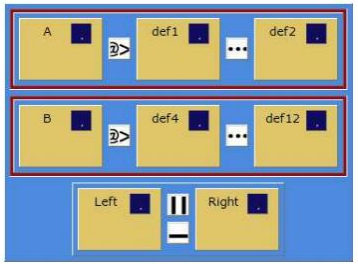
	Prefix	Prefix condition	Subject Prefix	Tense marker	Object prefix	Root	Verb ending
Present continuous	-		SUBJ_PREFIX	na	OBJ_PREFIX	ROOT	a
Simple past	-		SUBJ_PREFIX	li	OBJ_PREFIX	ROOT	a
Past perfect	-		SUBJ_PREFIX	me	OBJ_PREFIX	ROOT	a
Negative present			SUBJ_PREFIX	-	OBJ_PREFIX	ROOT	i
Negative past	si	Subject = 1SG		-	OBJ_PREFIX	ROOT	i
	hu	Subject = 2SG		-	OBJ_PREFIX	ROOT	i
	ha	Subject = 3SG					
	ha	(Elsewhere)	SUBJ_PREFIX	-	OBJ_PREFIX	ROOT	i

```
LEXICON TensePrefix

    !!! Affirmative Tenses (with mono ku-)
    [PRES][PROG]:@U.POL.POS@@U.KU.YES@na^          ObjectPrefix;
    [PAST]:@U.POL.POS@@U.KU.YES@li^                ObjectPrefix;
    [PAST][PERF]:@U.POL.POS@@U.KU.YES@me^          ObjectPrefix;
    [FUT]:@U.POL.POS@@U.KU.YES@ta^                 ObjectPrefix;
    [FUT]:@U.POL.POS@@U.KU.YES@ta^ka^              ObjectPrefix;
    [PRES][COND]:@U.POL.POS@@U.KU.YES@nge^         ObjectPrefix;
    [PAST][COND]:@U.POL.POS@@U.KU.YES@ngali^       ObjectPrefix;

    !!! Affirmative Tenses (without mono ku-)
    [PRES]:@U.POL.POS@@U.KU.NO@a^                  ObjectPrefix;
```

A visual programming language?

We originally were envisioning a visual programming language

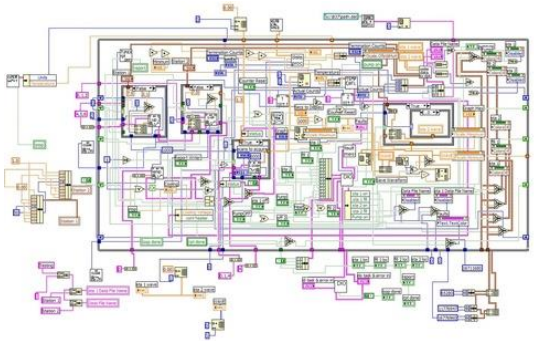
- Like Blockly (Fraser, 2015; Pasternak et al., 2017) for beginner programmers,
- Vi-xfst (Oflazer and Yilmaz, 2004) for FST development,
- or Galaxy-LAPPs (Ide et al., 2016)



Vi-XFST (Yılmaz, 2003)

But visual programming tends to shine for problems that can be solved by connecting 10-20 components.

- Some of our legacy products had ~10,000 components.
- Visual programming at that scale tends to have legibility problems.



Visual programming at scale…

Also, looking into the history of visual programming languages, we were concerned about their longevity.

- They often don't outlive their custom editors. Would we need to port the editor to new platforms forever?

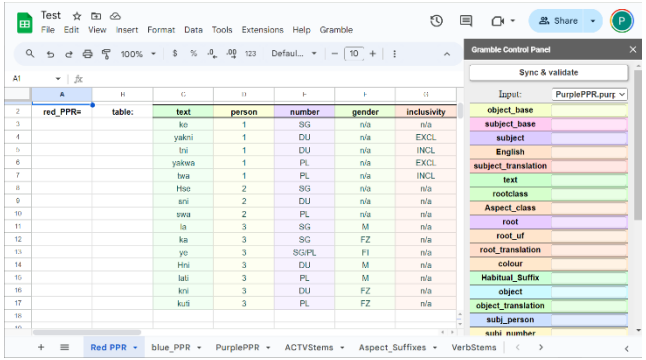
A tabular language

We soon realized that one “table” component would be make up >90% of the code anyway.

- Could we make it all tabular?
- This way, the SME’s spreadsheets and the programmer’s code could be the same document.

Also, spreadsheet editors are widely available and familiar.

- And have built-in sharing and live group editing.
- We made a plugin to Google Sheets that allows its use as an “IDE” for Gramble.
    - Note that Gramble source are just plain CSV files, and internally it does not depend on Google services. This is just a plugin to make editing Gramble files more ergonomic in that ecosystem



A tabular language

These features together greatly improved our productivity and ability to meaningfully involve collaborators.

- Concise tabular syntax similar to a spreadsheet/database.
- Live group programming in a familiar environment.
- Multi-tape regex engine underneath avoids awkward workarounds like @FLAGS@.
- Many small quality-of-life features for multi-person projects, like namespacing and built-in unit testing.

Gramble has been used to make products in 8 languages so far.

- Early adopters self-reported that they felt 10x more productive.
- Probably not true if we measured it with a stopwatch, but a testament to how much they preferred the new environment.
- One project has been successfully handed off from the original programmer to the SME, and has continued to grow and change.

Thanks!

This project would not have been possible without extensive testing, feedback, and brainstorming with our collaborators, coworkers, and the first generation of Gramble programmers:

- Including but not limited to Onkwawenna Kentyohkwa, the Prairies to Woodlands Indigenous Language Revitalization Circle, the Oneida Nation of the Thames, the WSÁNEĆ School Board, the Kitigan Zibi Cultural Education Center, Eddie Antonio Santos, Anna Kazantseva, Skylar Maguire, Kendra Hicks, Delaney Lothian, Akwiratékha' Martin, Yanfei Lu, and Michael Running Wolf.
- Funding for this project was provided by the National Research Council Ideation Small Teams grant *Speech Generation for Indigenous Language Education*.