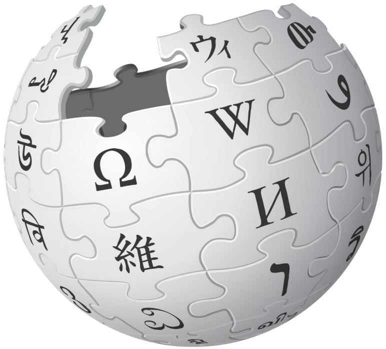
Abstract Wikipedia

and Vastly Multilingual Natural Language Generation



Aarne Ranta
CSE, Chalmers and GU
Digital Grammars AB

LACompLing, 17 December 2021

digital Grammars
Language technology to rely on.

Plan

The challenge

Grammatical Framework (GF)

NLG in GF: baseline and extensions

Research questions

The challenge

slides by Denny Vrandečić, Wikimedia

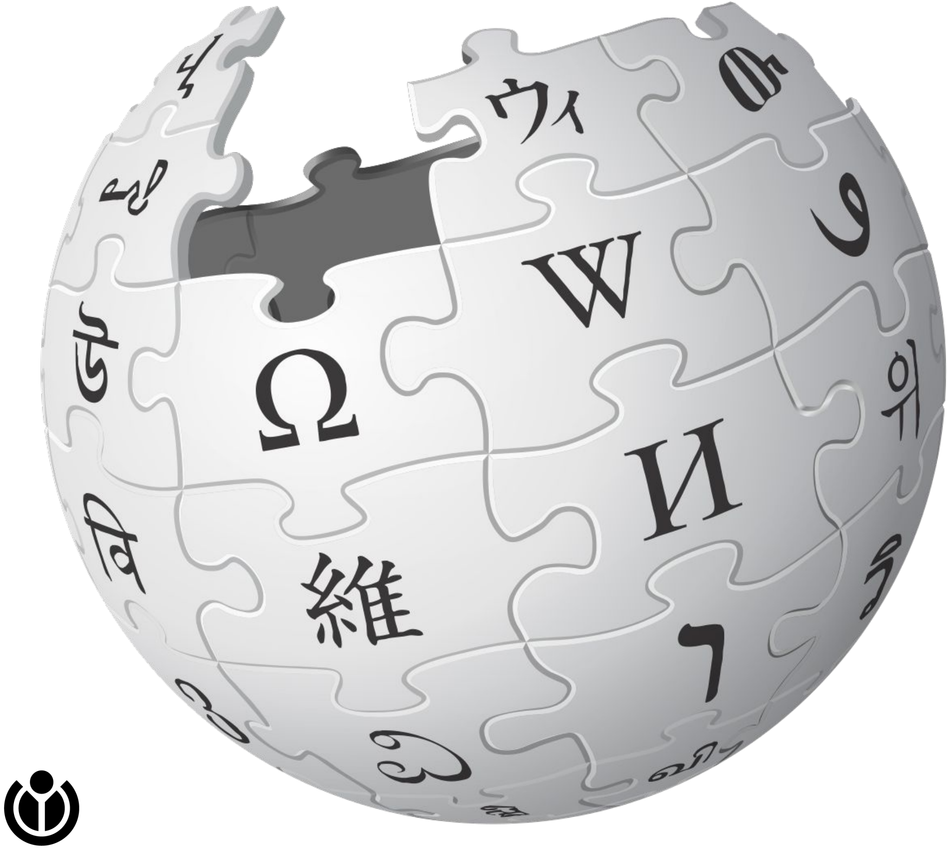
Imagine a world where everyone can share in the sum of all knowledge

WIKIMEDIA
FOUNDATION

Wikipedia today

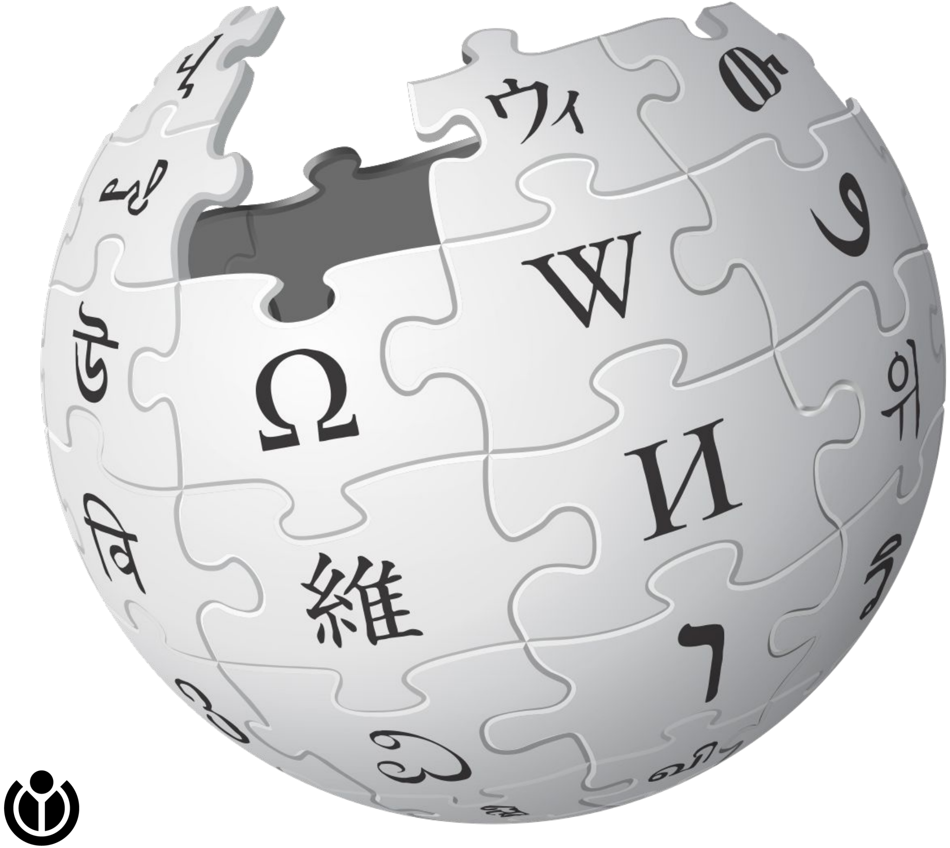
310 languages

56.3 million articles



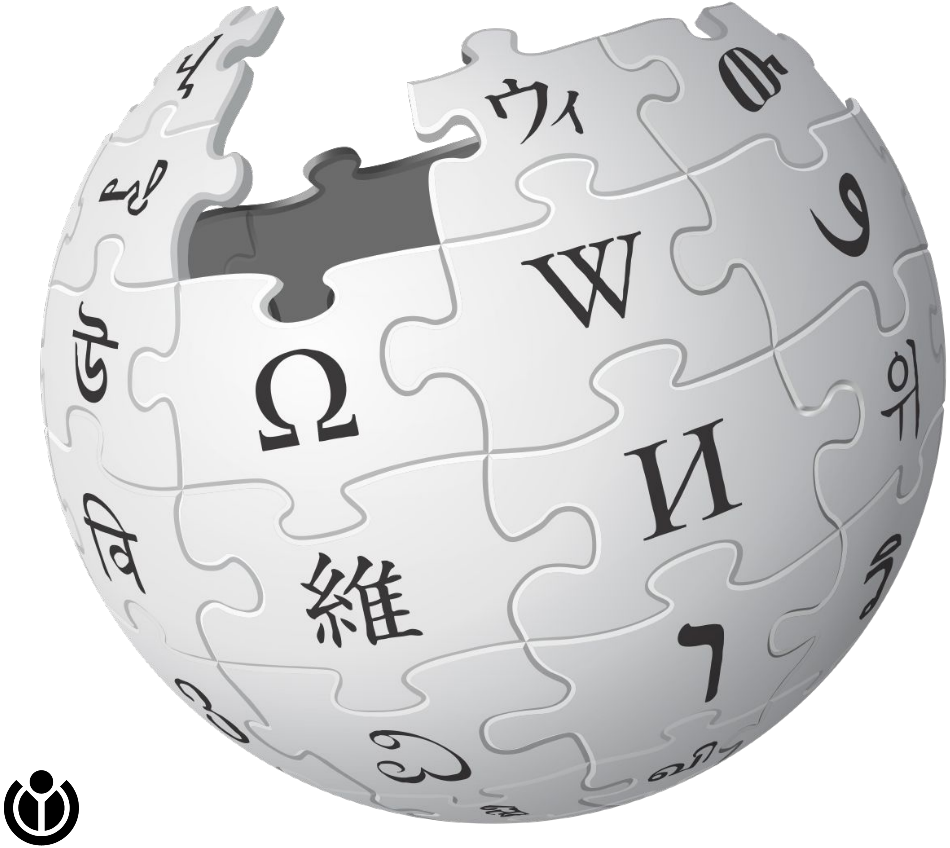
WIKIMEDIA FOUNDATION

Very unevenly distributed



Very unevenly distributed

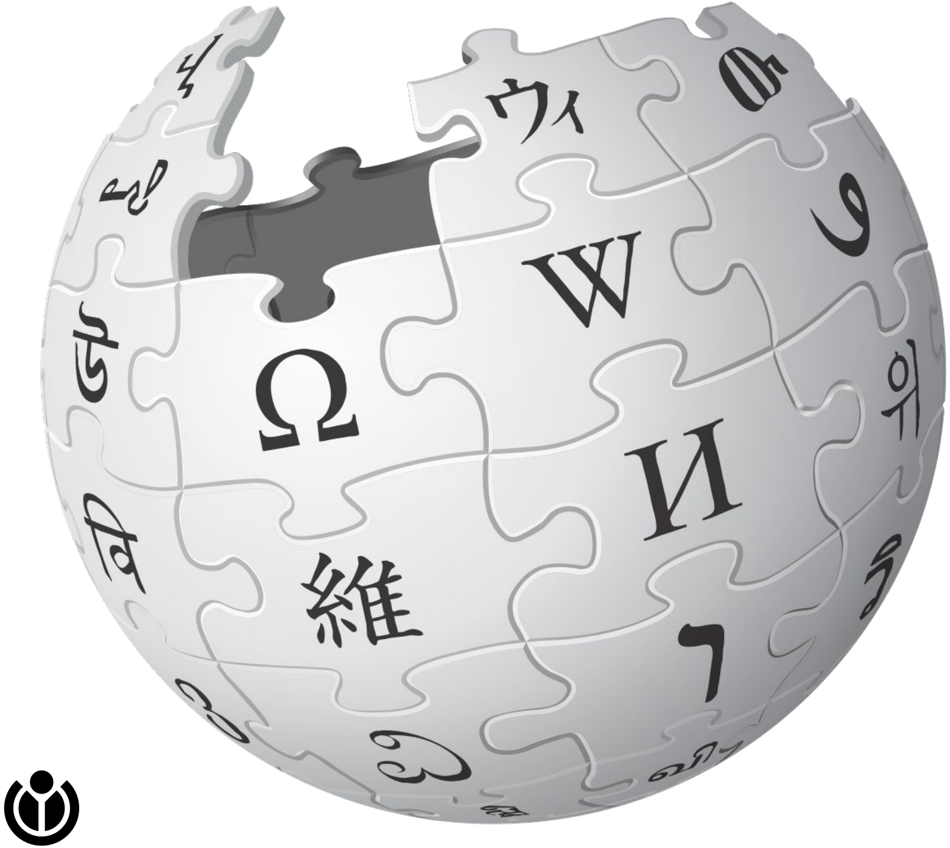
English: 6,316,595



Very unevenly distributed

English: 6,316,595

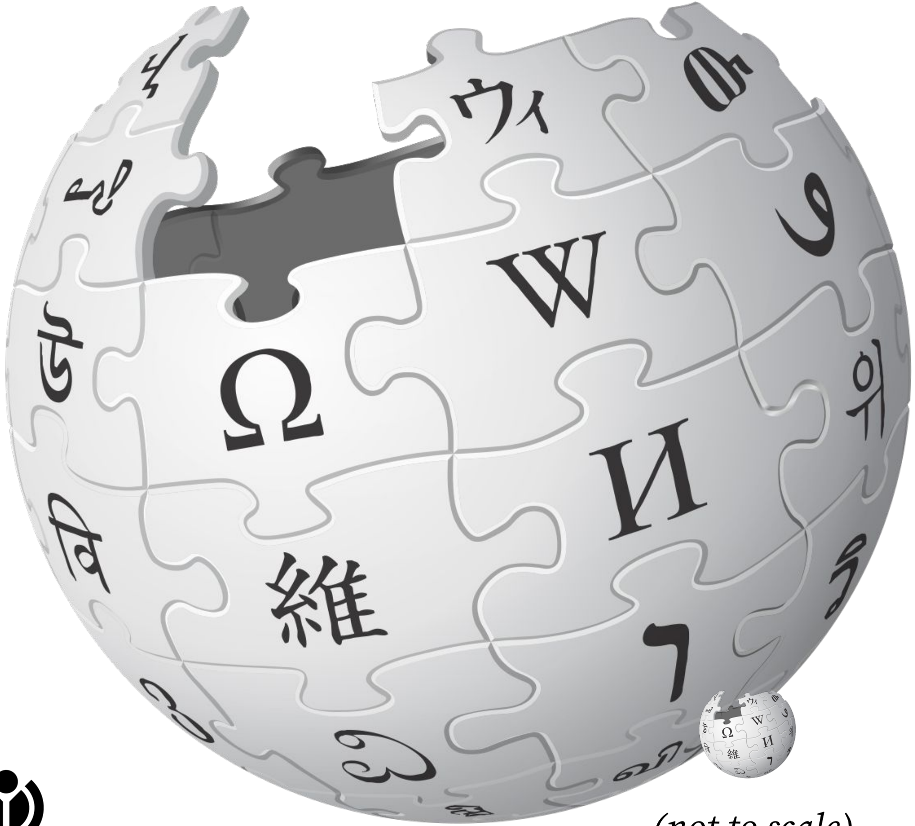
Amharic:



Very unevenly distributed

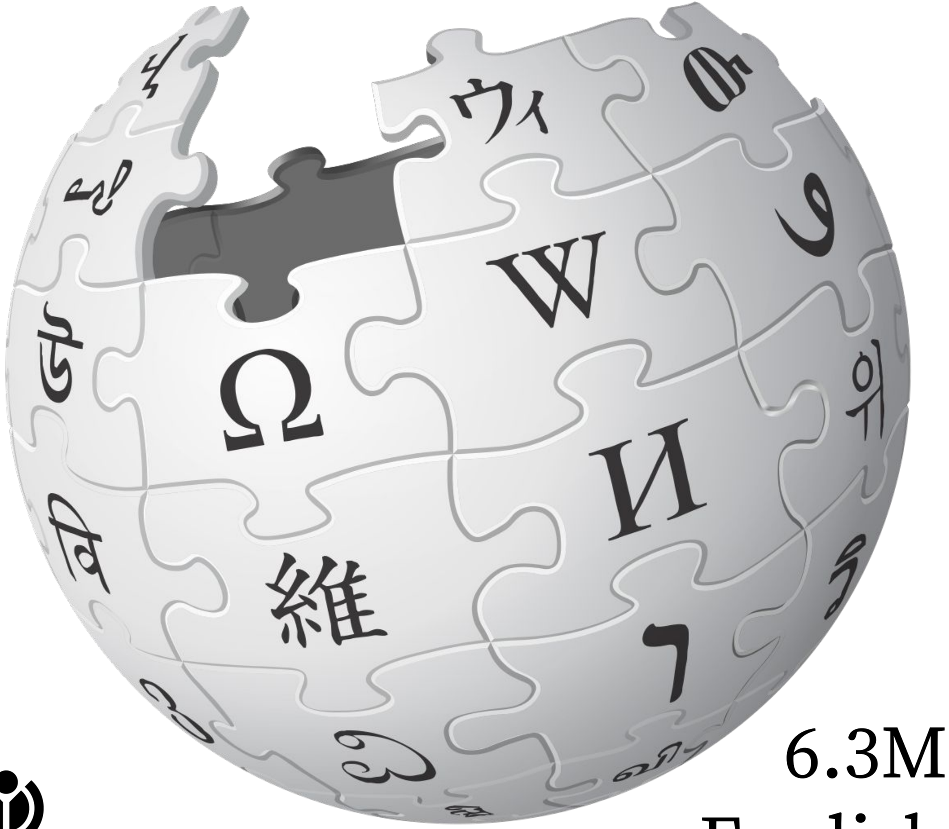
English: 6,316,595

Amharic: 14,941



(not to scale)

WIKIMEDIA FOUNDATION



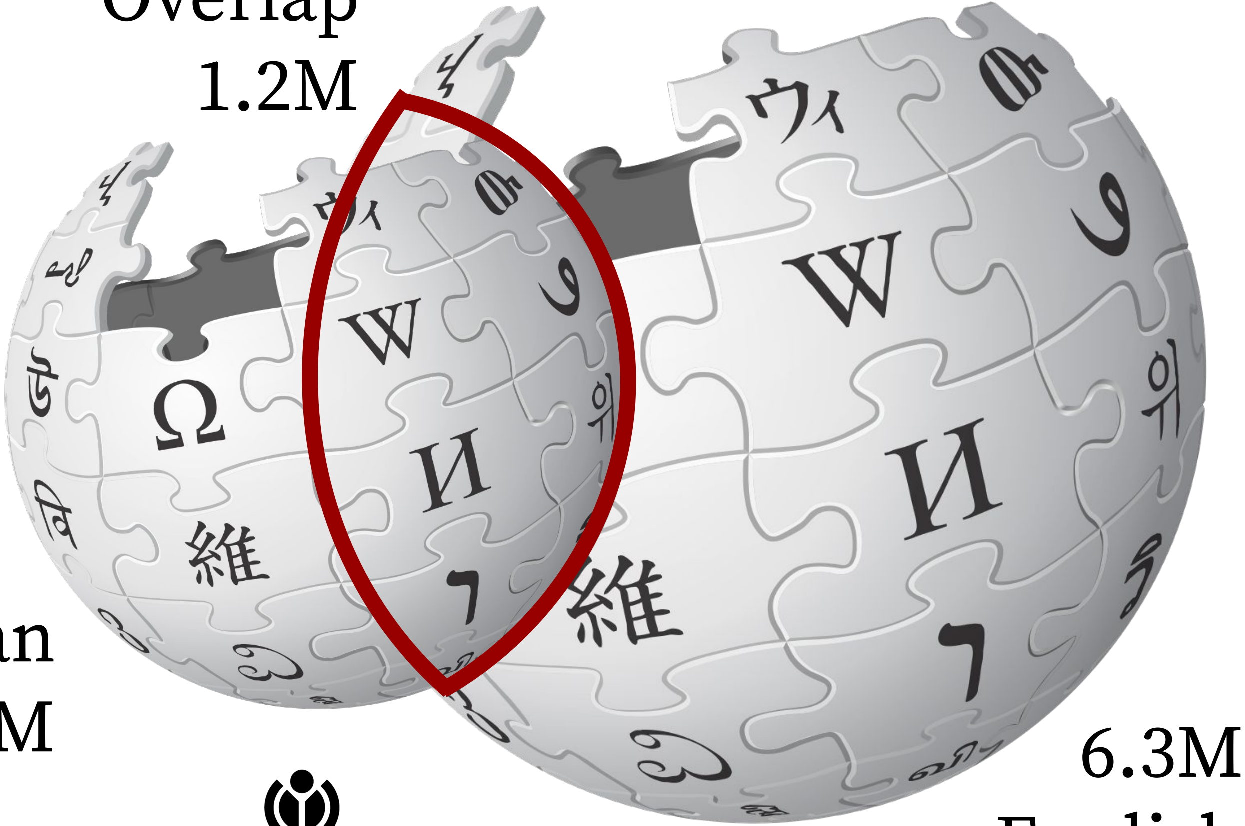
WIKIMEDIA
FOUNDATION

6.3M
English

German
2.6M

6.3M
English

WIKIMEDIA
FOUNDATION



Overlap
1.2M

German
2.6M

6.3M
English

WIKIMEDIA
FOUNDATION

Coverage

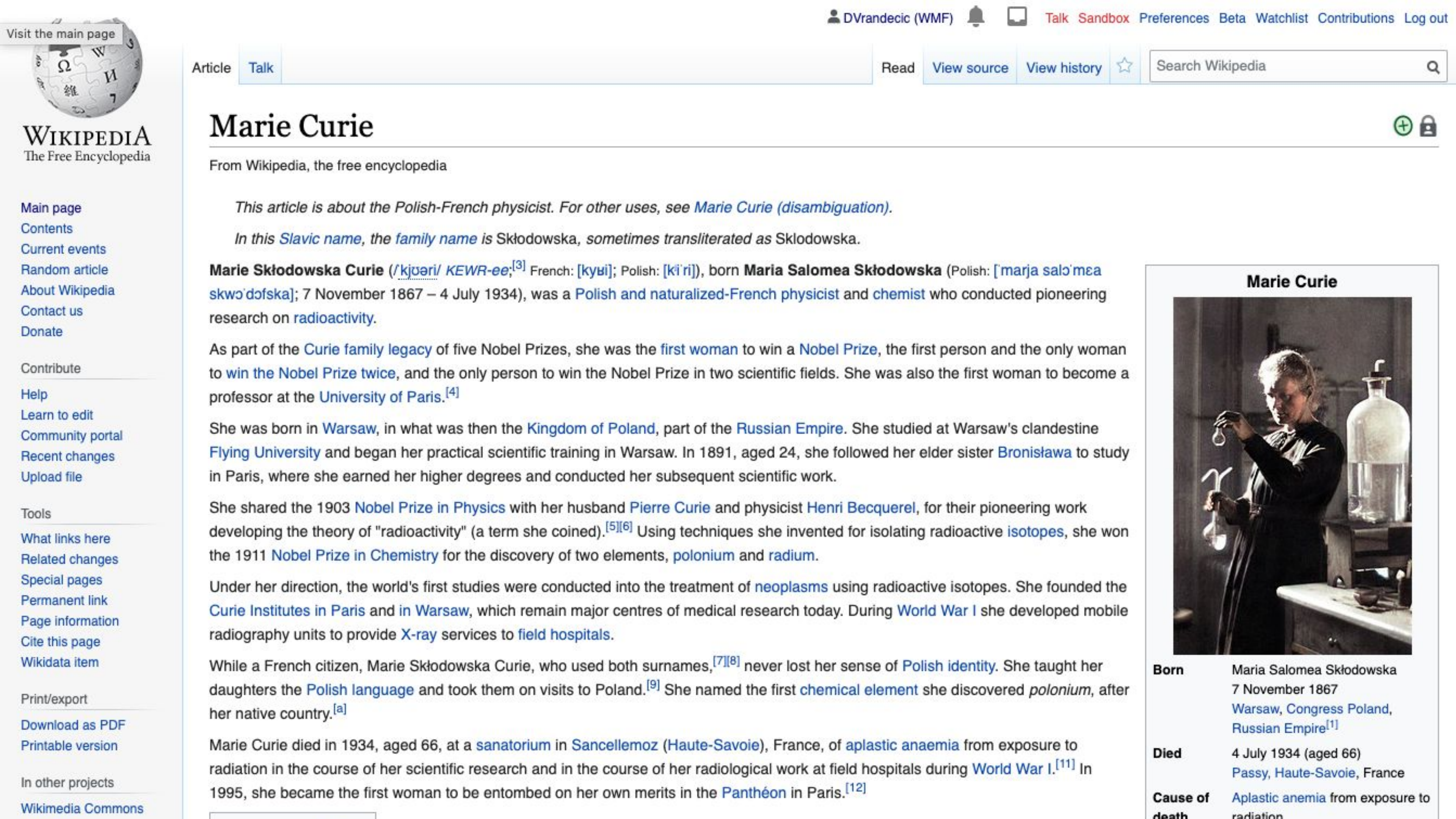
20M topics with Wikipedia articles

WIKIMEDIA
FOUNDATION

Coverage

20M topics with Wikipedia articles

English covers 6M (~⅓)



Marie Curie

From Wikipedia, the free encyclopedia

This article is about the Polish-French physicist. For other uses, see Marie Curie (disambiguation).

In this Slavic name, the family name is Skłodowska, *sometimes transliterated as* Sklodowska.

Marie Skłodowska Curie (/ˈkjʊəri/ *KEWR-ee*;[3] French: [kyʁi]; Polish: [kʲiˈri]), born **Maria Salomea Skłodowska** (Polish: [ˈmarja salɔˈmɛa skwɔˈdɔfska]; 7 November 1867 – 4 July 1934), was a Polish and naturalized-French physicist and chemist who conducted pioneering research on radioactivity.

As part of the Curie family legacy of five Nobel Prizes, she was the first woman to win a Nobel Prize, the first person and the only woman to win the Nobel Prize twice, and the only person to win the Nobel Prize in two scientific fields. She was also the first woman to become a professor at the University of Paris.[4]

She was born in Warsaw, in what was then the Kingdom of Poland, part of the Russian Empire. She studied at Warsaw's clandestine Flying University and began her practical scientific training in Warsaw. In 1891, aged 24, she followed her elder sister Bronisława to study in Paris, where she earned her higher degrees and conducted her subsequent scientific work.

She shared the 1903 Nobel Prize in Physics with her husband Pierre Curie and physicist Henri Becquerel, for their pioneering work developing the theory of "radioactivity" (a term she coined).[5][6] Using techniques she invented for isolating radioactive isotopes, she won the 1911 Nobel Prize in Chemistry for the discovery of two elements, polonium and radium.

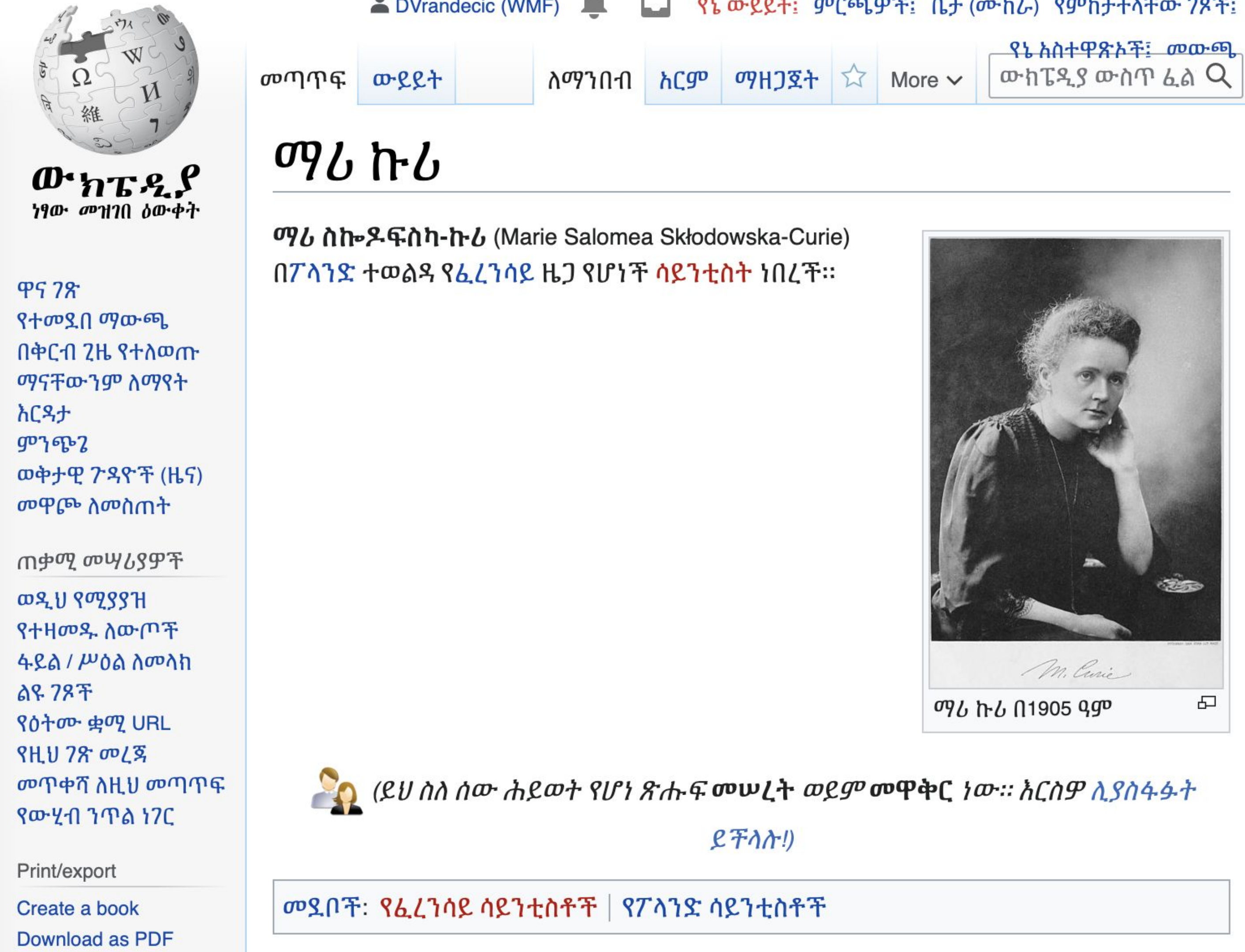
Under her direction, the world's first studies were conducted into the treatment of neoplasms using radioactive isotopes. She founded the Curie Institutes in Paris and in Warsaw, which remain major centres of medical research today. During World War I she developed mobile radiography units to provide X-ray services to field hospitals.

While a French citizen, Marie Skłodowska Curie, who used both surnames,[7][8] never lost her sense of Polish identity. She taught her daughters the Polish language and took them on visits to Poland.[9] She named the first chemical element she discovered *polonium*, after her native country.[a]

Marie Curie died in 1934, aged 66, at a sanatorium in Sancellemoz (Haute-Savoie), France, of aplastic anaemia from exposure to radiation in the course of her scientific research and in the course of her radiological work at field hospitals during World War I.[11] In 1995, she became the first woman to be entombed on her own merits in the Panthéon in Paris.[12]

Marie Curie

Born	Maria Salomea Skłodowska 7 November 1867 Warsaw, Congress Poland, Russian Empire[1]
Died	4 July 1934 (aged 66) Passy, Haute-Savoie, France
Cause of death	Aplastic anemia from exposure to radiation



DVrandecic (WMF) የኔ ውይይት፤ ምርጫዎች፤ ቤታ (ሙከራ) የምከታተላቸው ገጾች፤ የኔ አስተዋጽኦች፤ መውጫ

መጣጥፍ ውይይት ለማንበብ አርም ማዘጋጀት ☆ More ⌄ ውክፔዲያ ውስጥ ፈል

ውክፔዲያ
ነፃው መዝገበ ዕውቀት

- ዋና ገጽ
- የተመደበ ማውጫ
- በቅርብ ጊዜ የተለወጡ
- ማናቸውንም ለማየት
- እርዳታ
- ምንጭጌ
- ወቅታዊ ጉዳዮች (ዜና)
- መዋጮ ለመስጠት

ጠቃሚ መሣሪያዎች

- ወዲህ የሚያያዝ
- የተዛመዱ ለውጦች
- ፋይል / ሥዕል ለመላክ
- ልዩ ገጾች
- የዕትሙ ቋሚ URL
- የዚህ ገጽ መረጃ
- መጥቀሻ ለዚህ መጣጥፍ
- የውሂብ ንጥል ነገር

Print/export

- Create a book
- Download as PDF

ማሪ ኩሪ

ማሪ ስኰዶፍስካ-ኩሪ (Marie Salomea Skłodowska-Curie) በፖላንድ ተወልዳ የፈረንሳይ ዜጋ የሆነች ሳይንቲስት ነበረች።

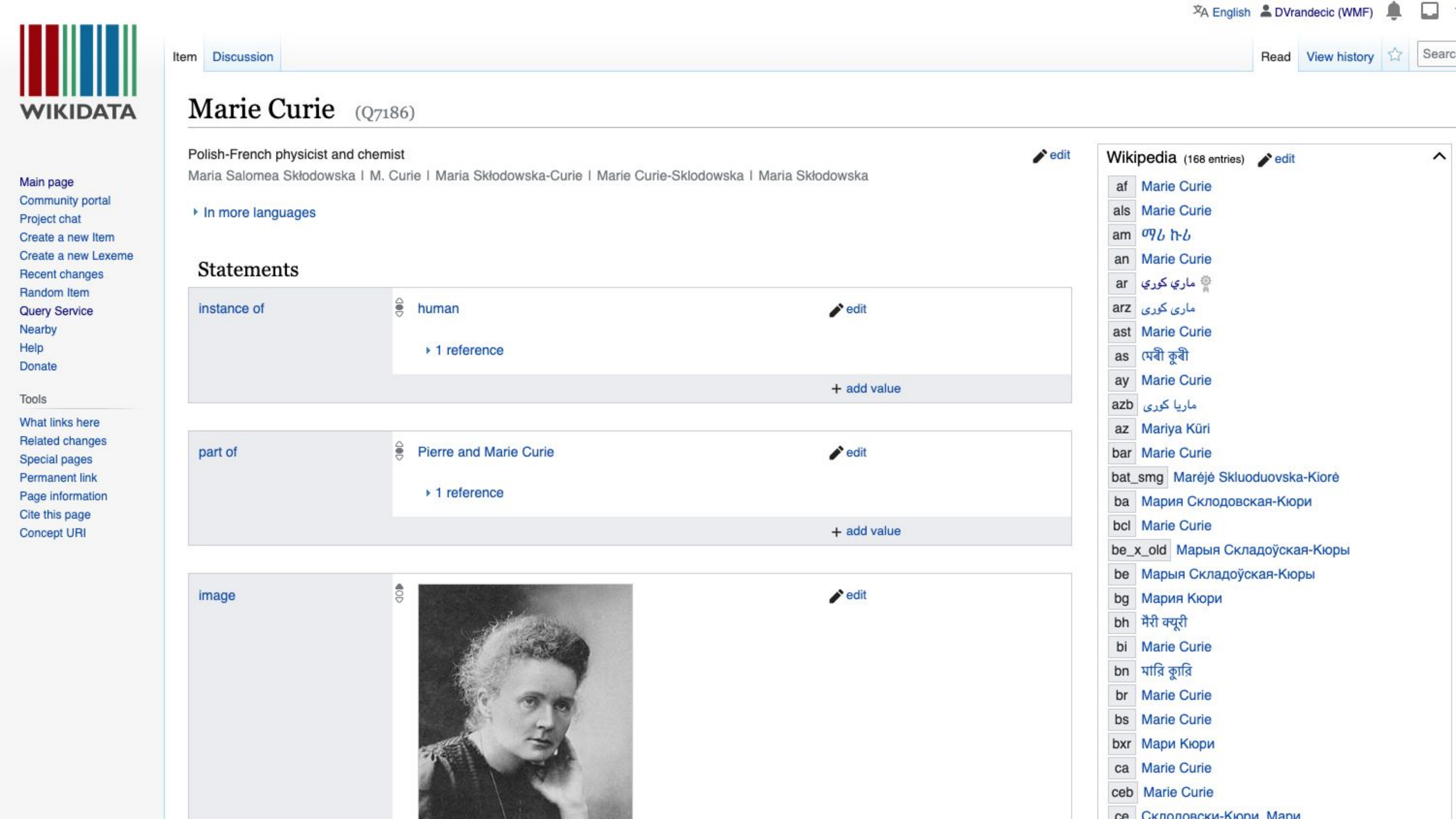
ማሪ ኩሪ በ1905 ዓም

(ይህ ስለ ሰው ሕይወት የሆነ ጽሑፍ ***መሠረት*** *ወይም* ***መዋቅር*** *ነው። እርስዎ ሊያስፋፉት ይችላሉ!)*

መደቦች: የፈረንሳይ ሳይንቲስቶች | የፖላንድ ሳይንቲስቶች

WIKIDATA

WIKIMEDIA
FOUNDATION



文A English DVrandecic (WMF)

WIKIDATA

Item Discussion Read View history ☆ Searc

Main page
Community portal
Project chat
Create a new Item
Create a new Lexeme
Recent changes
Random Item
Query Service
Nearby
Help
Donate

Tools

What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Concept URI

Marie Curie (Q7186)

Polish-French physicist and chemist edit

Maria Salomea Skłodowska | M. Curie | Maria Skłodowska-Curie | Marie Curie-Sklodowska | Maria Skłodowska

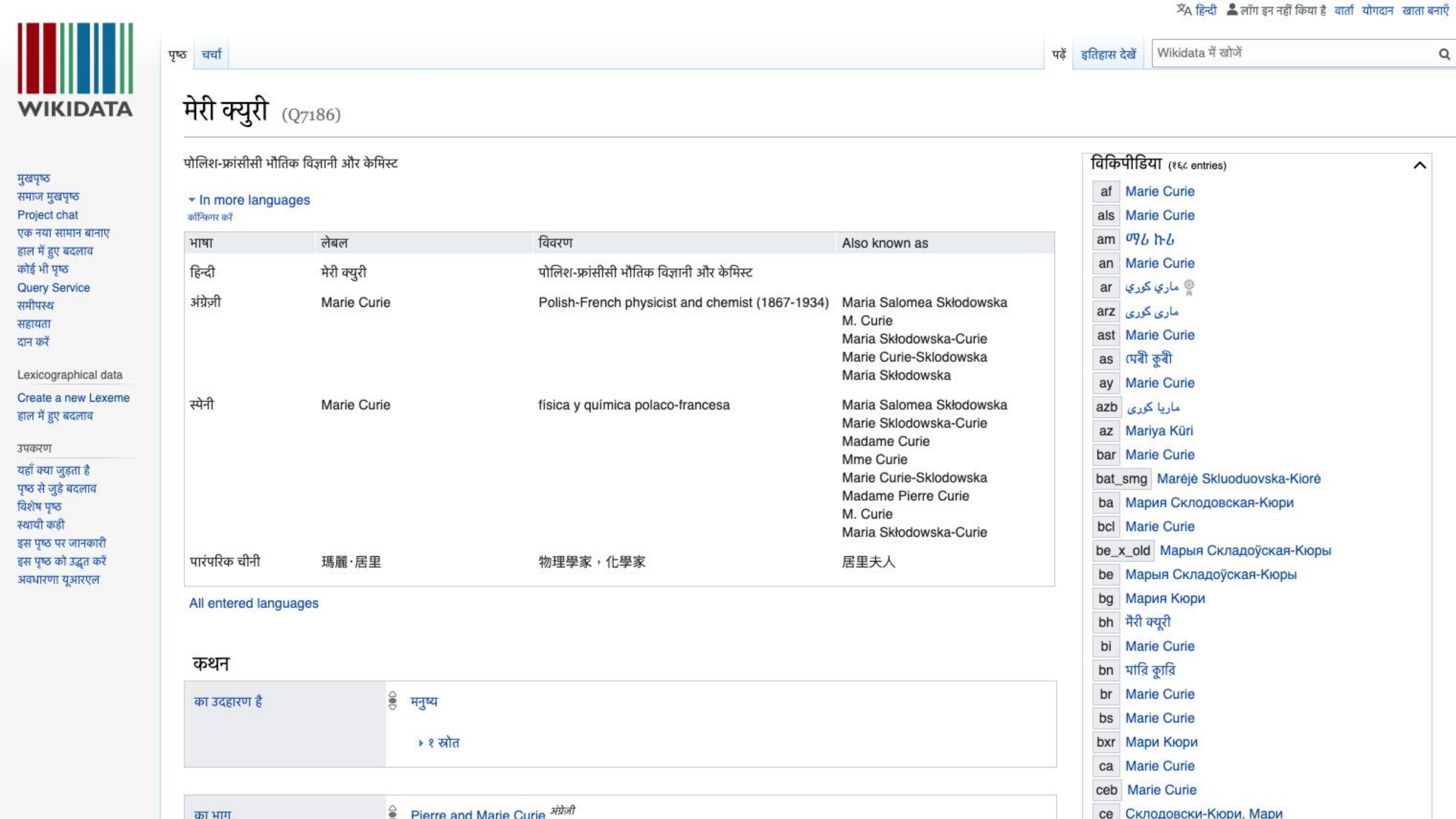
▸ In more languages

Statements

instance of	human ▸ 1 reference + add value	edit
part of	Pierre and Marie Curie ▸ 1 reference + add value	edit
image		edit

Wikipedia (168 entries) edit

af	Marie Curie
als	Marie Curie
am	ማሪ ኩሪ
an	Marie Curie
ar	ماري كوري
arz	مارى كورى
ast	Marie Curie
as	মেৰী কুৰী
ay	Marie Curie
azb	ماریا کوری
az	Mariya Küri
bar	Marie Curie
bat_smg	Marėjė Skluoduovska-Kiorė
ba	Мария Склодовская-Кюри
bcl	Marie Curie
be_x_old	Марыя Складоўская-Кюры
be	Марыя Складоўская-Кюры
bg	Мария Кюри
bh	मैरी क्यूरी
bi	Marie Curie
bn	মারি কুরি
br	Marie Curie
bs	Marie Curie
bxr	Мари Кюри
ca	Marie Curie
ceb	Marie Curie
ce	Склодовски-Кюри, Мари



मेरी क्युरी (Q7186)

पोलिश-फ्रांसीसी भौतिक विज्ञानी और केमिस्ट

In more languages

कॉन्फिगर करें

भाषा	लेबल	विवरण	Also known as
हिन्दी	मेरी क्युरी	पोलिश-फ्रांसीसी भौतिक विज्ञानी और केमिस्ट	
अंग्रेज़ी	Marie Curie	Polish-French physicist and chemist (1867-1934)	Maria Salomea Skłodowska M. Curie Maria Skłodowska-Curie Marie Curie-Sklodowska Maria Skłodowska
स्पेनी	Marie Curie	física y química polaco-francesa	Maria Salomea Skłodowska Marie Sklodowska-Curie Madame Curie Mme Curie Marie Curie-Sklodowska Madame Pierre Curie M. Curie Maria Skłodowska-Curie
पारंपरिक चीनी	瑪麗·居里	物理學家，化學家	居里夫人

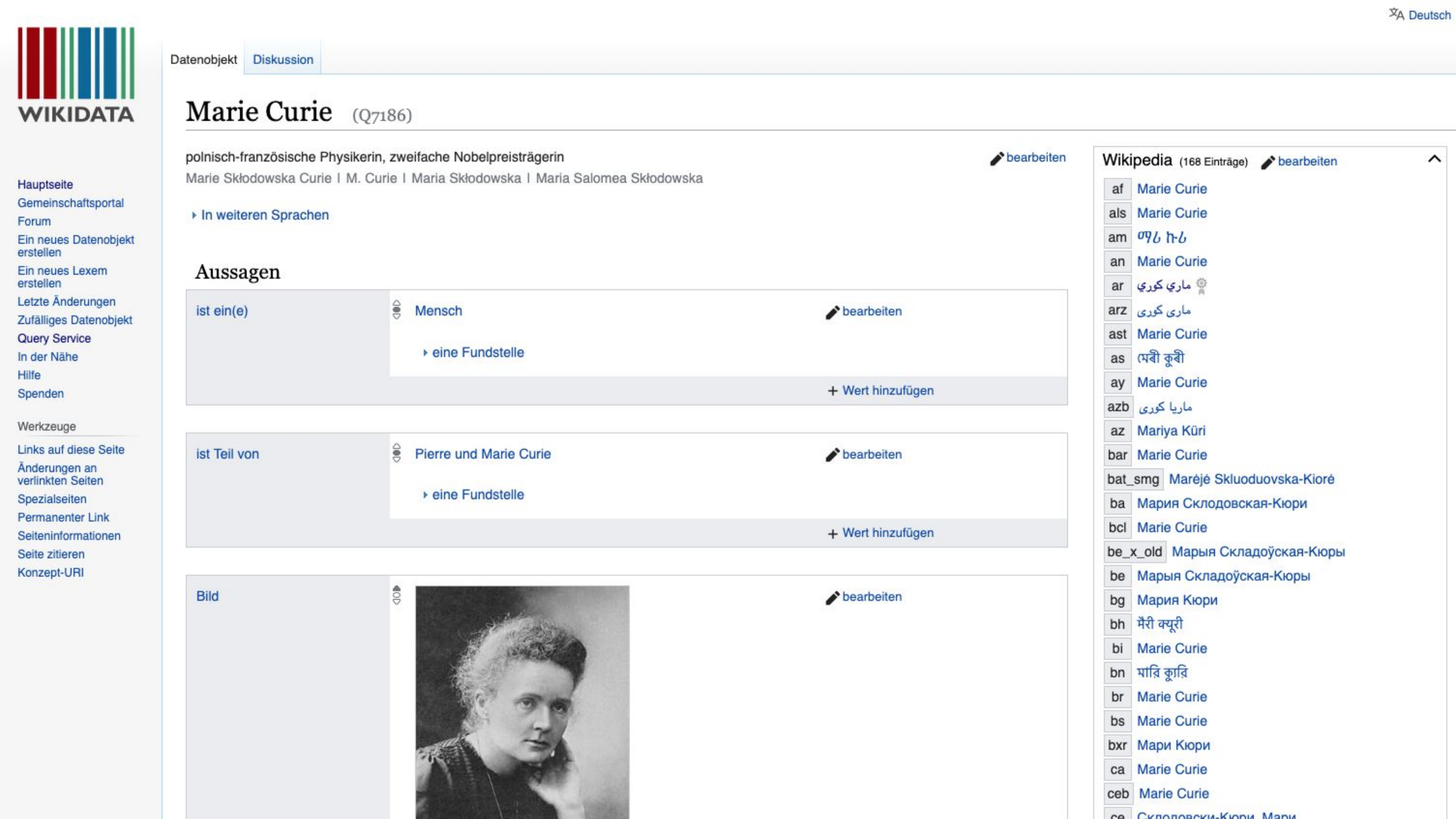
All entered languages

कथन

का उदाहरण है	मनुष्य १ स्रोत
का भाग	Pierre and Marie Curie अंग्रेज़ी

विकिपीडिया (१६८ entries)

af	Marie Curie
als	Marie Curie
am	ማሪ ኩሪ
an	Marie Curie
ar	ماري كوري
arz	مارى كورى
ast	Marie Curie
as	মেৰী কুৰী
ay	Marie Curie
azb	ماریا کوری
az	Mariya Küri
bar	Marie Curie
bat_smg	Marėjė Skluoduovska-Kiorė
ba	Мария Склодовская-Кюри
bcl	Marie Curie
be_x_old	Марыя Складоўская-Кюры
be	Марыя Складоўская-Кюры
bg	Мария Кюри
bh	मैरी क्यूरी
bi	Marie Curie
bn	মারি ক্যুরি
br	Marie Curie
bs	Marie Curie
bxr	Мари Кюри
ca	Marie Curie
ceb	Marie Curie
ce	Склодовски-Кюри, Мари



文A Deutsch

WIKIDATA

Datenobjekt | Diskussion

Marie Curie (Q7186)

polnisch-französische Physikerin, zweifache Nobelpreisträgerin

Marie Skłodowska Curie | M. Curie | Maria Skłodowska | Maria Salomea Skłodowska

bearbeiten

‣ In weiteren Sprachen

Aussagen

ist ein(e)	Mensch ‣ eine Fundstelle	bearbeiten
		+ Wert hinzufügen
ist Teil von	Pierre und Marie Curie ‣ eine Fundstelle	bearbeiten
		+ Wert hinzufügen
Bild		bearbeiten

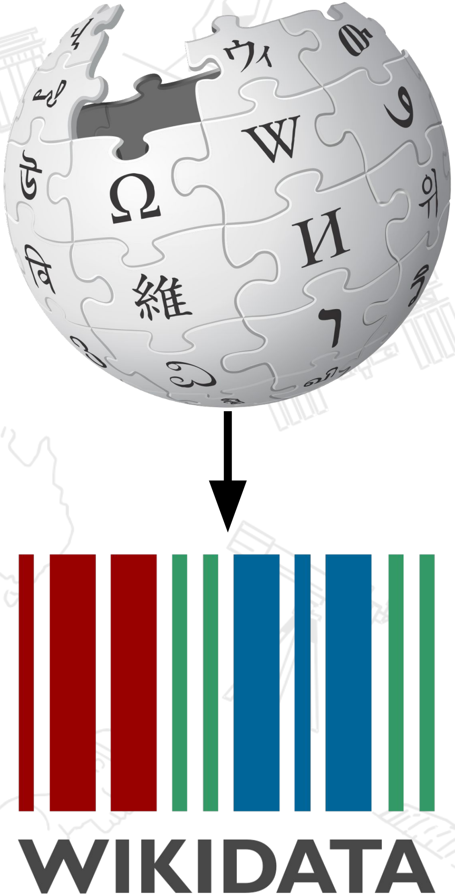
Hauptseite
Gemeinschaftsportal
Forum
Ein neues Datenobjekt erstellen
Ein neues Lexem erstellen
Letzte Änderungen
Zufälliges Datenobjekt
Query Service
In der Nähe
Hilfe
Spenden

Werkzeuge

Links auf diese Seite
Änderungen an verlinkten Seiten
Spezialseiten
Permanenter Link
Seiteninformationen
Seite zitieren
Konzept-URI

Wikipedia (168 Einträge) bearbeiten

af	Marie Curie
als	Marie Curie
am	ማሪ ኩሪ
an	Marie Curie
ar	ماري كوري
arz	مارى كورى
ast	Marie Curie
as	মেৰী কুৰী
ay	Marie Curie
azb	ماریا کوری
az	Mariya Küri
bar	Marie Curie
bat_smg	Marėjė Skluoduovska-Kiorė
ba	Мария Склодовская-Кюри
bcl	Marie Curie
be_x_old	Марыя Складоўская-Кюры
be	Марыя Складоўская-Кюры
bg	Мария Кюри
bh	मैरी क्यूरी
bi	Marie Curie
bn	মারি কুরি
br	Marie Curie
bs	Marie Curie
bxr	Мари Кюри
ca	Marie Curie
ceb	Marie Curie
ce	Склодовски-Кюри, Мари



Just bring everything from Wikipedia to Wikidata

WIKIMEDIA
FOUNDATION

The cost of Wikipedia

#topics * #languages

20M * 300 ≈ ~~6~~2,000,000,0~~00~~

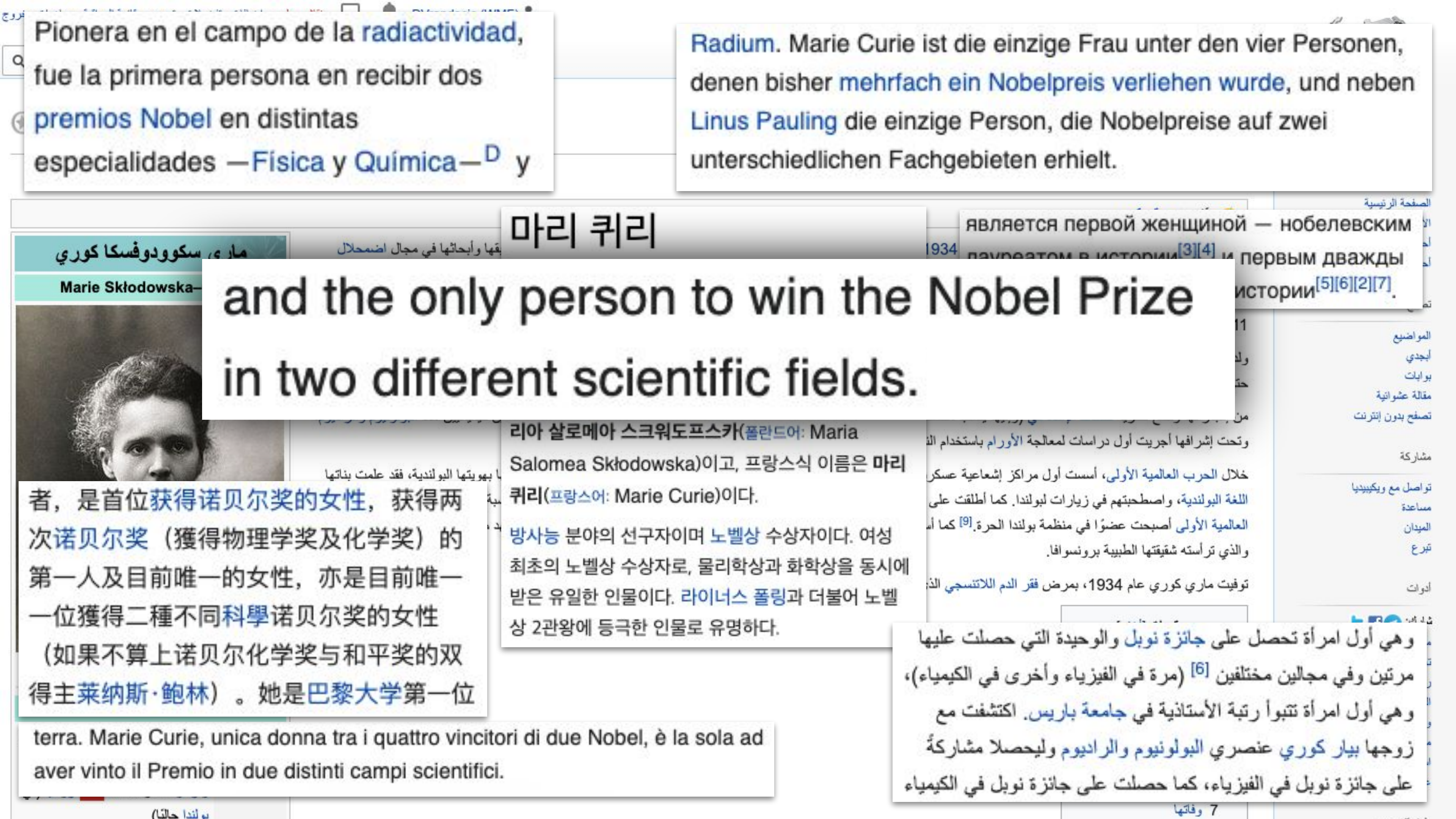
WIKIMEDIA
FOUNDATION

93,923,690 items

WIKIMEDIA
FOUNDATION

1,283,552,937 statements

WIKIMEDIA
FOUNDATION



Pionera en el campo de la radiactividad, fue la primera persona en recibir dos premios Nobel en distintas especialidades —Física y Química—[D] y

Radium. Marie Curie ist die einzige Frau unter den vier Personen, denen bisher mehrfach ein Nobelpreis verliehen wurde, und neben Linus Pauling die einzige Person, die Nobelpreise auf zwei unterschiedlichen Fachgebieten erhielt.

마리 퀴리

является первой женщиной — нобелевским лауреатом в истории[3][4] и первым дважды

истории[5][6][2][7].

ماري سكوودوفسكا كوري

Marie Skłodowska–

and the only person to win the Nobel Prize in two different scientific fields.

者，是首位获得诺贝尔奖的女性，获得两次诺贝尔奖（獲得物理学奖及化学奖）的第一人及目前唯一的女性，亦是目前唯一一位獲得二種不同科學诺贝尔奖的女性（如果不算上诺贝尔化学奖与和平奖的双得主莱纳斯·鲍林）。她是巴黎大学第一位

리아 살로메아 스크워도프스카(폴란드어: Maria Salomea Skłodowska)이고, 프랑스식 이름은 **마리 퀴리**(프랑스어: Marie Curie)이다.

방사능 분야의 선구자이며 노벨상 수상자이다. 여성 최초의 노벨상 수상자로, 물리학상과 화학상을 동시에 받은 유일한 인물이다. 라이너스 폴링과 더불어 노벨상 2관왕에 등극한 인물로 유명하다.

وهي أول امرأة تحصل على جائزة نوبل والوحيدة التي حصلت عليها
مرتين وفي مجالين مختلفين [6] (مرة في الفيزياء وأخرى في الكيمياء)،
وهي أول امرأة تتبوأ رتبة الأستاذية في جامعة باريس. اكتشفت مع
زوجها بيار كوري عنصري البولونيوم والراديوم وليحصلا مشاركة
على جائزة نوبل في الفيزياء، كما حصلت على جائزة نوبل في الكيمياء

terra. Marie Curie, unica donna tra i quattro vincitori di due Nobel, è la sola ad aver vinto il Premio in due distinti campi scientifici.

Toy example

```
OnlyPersonThat
  Person: Marie Curie
  Condition: AwardWinning
    Award: Nobel Prize
    Type: ModNounPhrase
      Determiner: two
      Modifiers:
        different
        scientific
      Head: category
```

“Marie Curie was the only person to receive the Nobel Prize in two different scientific categories.”

Toy example

```
OnlyPersonThat
  Person: Marie Curie
  Condition: AwardWinning
    Award: Nobel Prize
    Type: ModNounPhrase
      Determiner: two
      Modifiers:
        different
        scientific
      Head: category
```

“Marie Curie je jedina osoba koja je dobila Nobelovu nagradu u dva različita znanstvena područja.”

Constructors

```
OnlyPersonThat(
  Person: Person
  Condition: Verb phrase
) → Clause

AwardWinning(
  Award: Award
  Type: Noun phrase
  Time: Time
  Reason: Noun phrase
) → Verb phrase

ModNounPhrase(
  Determiner: Determiner
  Modifiers: List(Adjectival phrase)
  Head: Noun phrase
) → Noun phrase
```

```
OnlyPersonThat:
  Person +
  "was the only person" +
  Condition + "."
```

Renderer

per constructor and language

"Marie Curie ***was the only person who*** *to receive the Nobel Prize in two different scientific categories."*

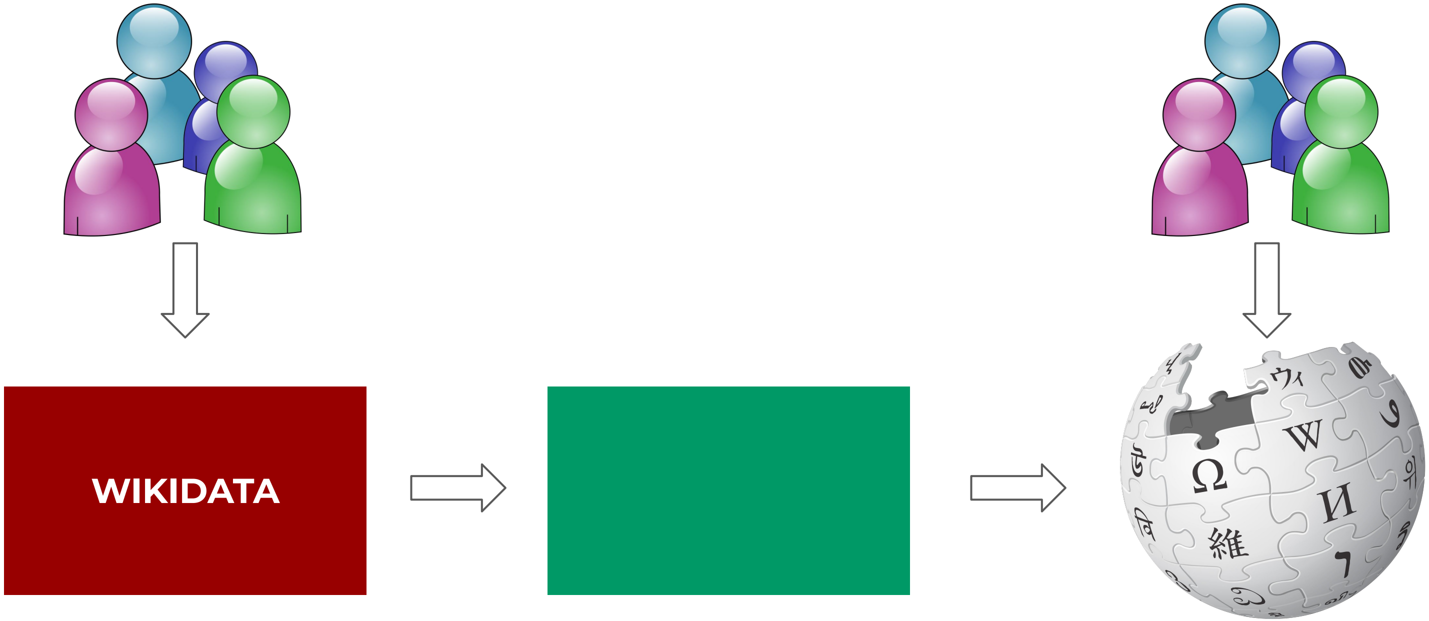
```
OnlyPersonThat:
  Person +
  "was the only person" +
  Condition + "."
```

```
OnlyPersonThat:
  Person +
  "je jedina osoba koja je"
  + Condition + "."
```

Renderer

per constructor
and language

*“Marie Curie **je jedina osoba koja je** dobila Nobelovu nagradu u dva različita znanstvena područja.”*

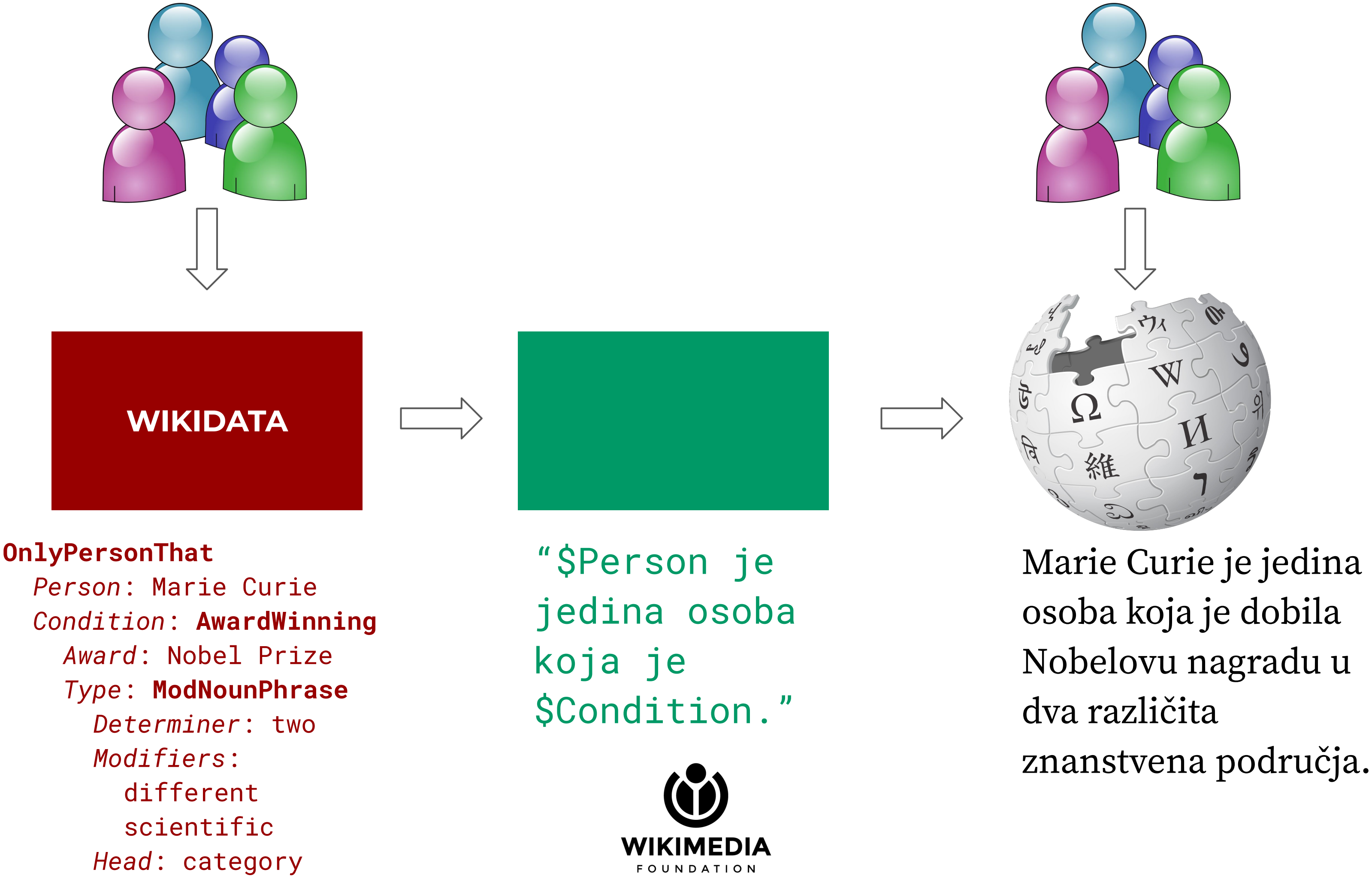


```
OnlyPersonThat
  Person: Marie Curie
  Condition: AwardWinning
    Award: Nobel Prize
    Type: ModNounPhrase
      Determiner: two
      Modifiers:
        different
        scientific
      Head: category
```

“$Person was the only person $Condition.”

WIKIMEDIA
FOUNDATION

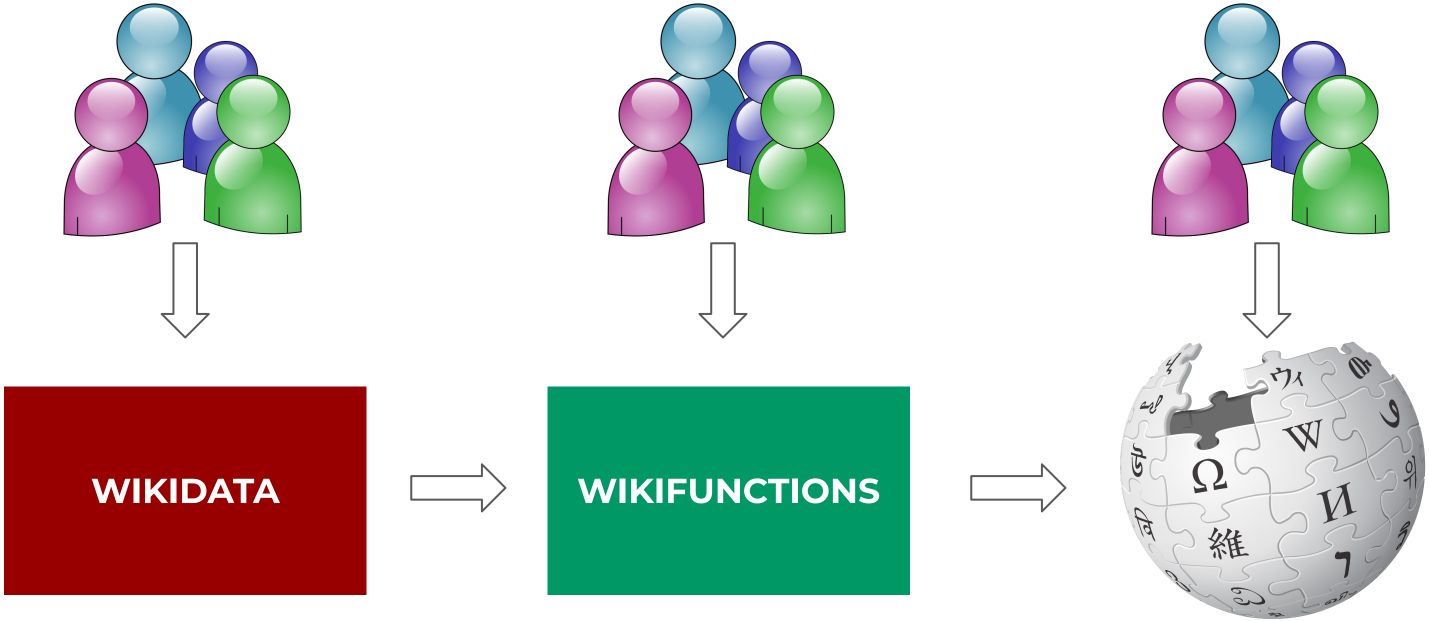
Marie Curie was the only person to receive the Nobel Prize in two different scientific categories.



```
OnlyPersonThat
  Person: Marie Curie
  Condition: AwardWinning
    Award: Nobel Prize
    Type: ModNounPhrase
      Determiner: two
      Modifiers:
        different
        scientific
      Head: category
```

“$Person je jedina osoba koja je $Condition.”

Marie Curie je jedina osoba koja je dobila Nobelovu nagradu u dva različita znanstvena područja.



WIKIDATA

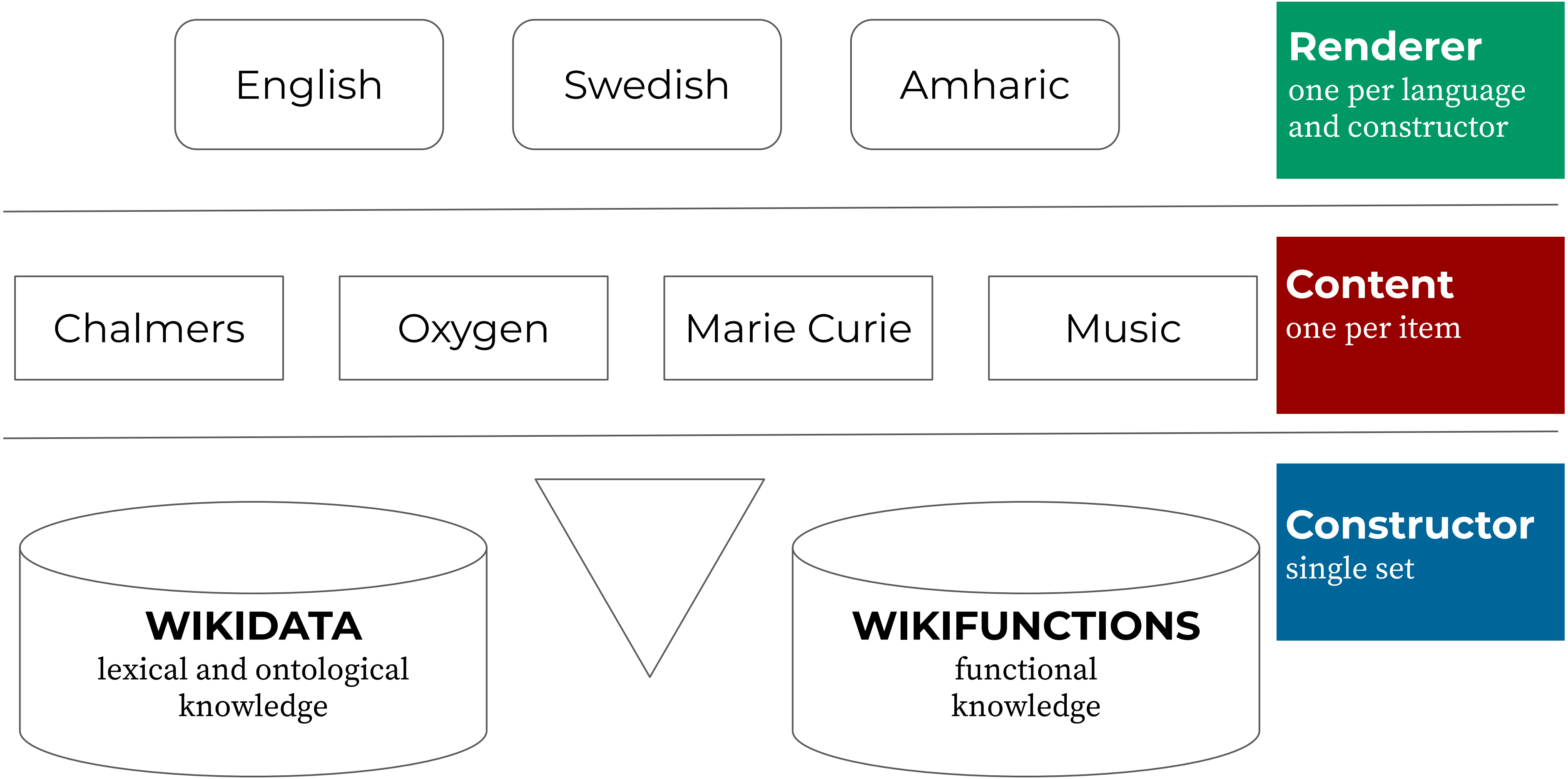
WIKIFUNCTIONS

```
OnlyPersonThat
  Person: Marie Curie
  Condition: AwardWinning
    Award: Nobel Prize
    Type: ModNounPhrase
      Determiner: two
      Modifiers:
        different
        scientific
      Head: category
```

“$Person je jedina osoba koja je $Condition.”

WIKIMEDIA
FOUNDATION

Marie Curie je jedina osoba koja je dobila Nobelovu nagradu u dva različita znanstvena područja.



English

Swedish

Amharic

Renderer
one per language
and constructor

Chalmers

Oxygen

Marie Curie

Music

Content
one per item

WIKIDATA
lexical and ontological
knowledge

WIKIFUNCTIONS
functional
knowledge

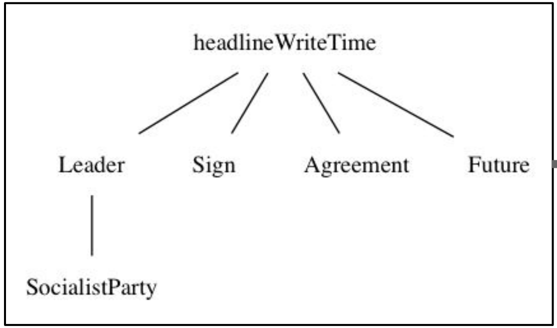
Constructor
single set

Grammatical Framework (GF)

The mission of GF is to formalize the grammars of the world and make them available for computer applications.

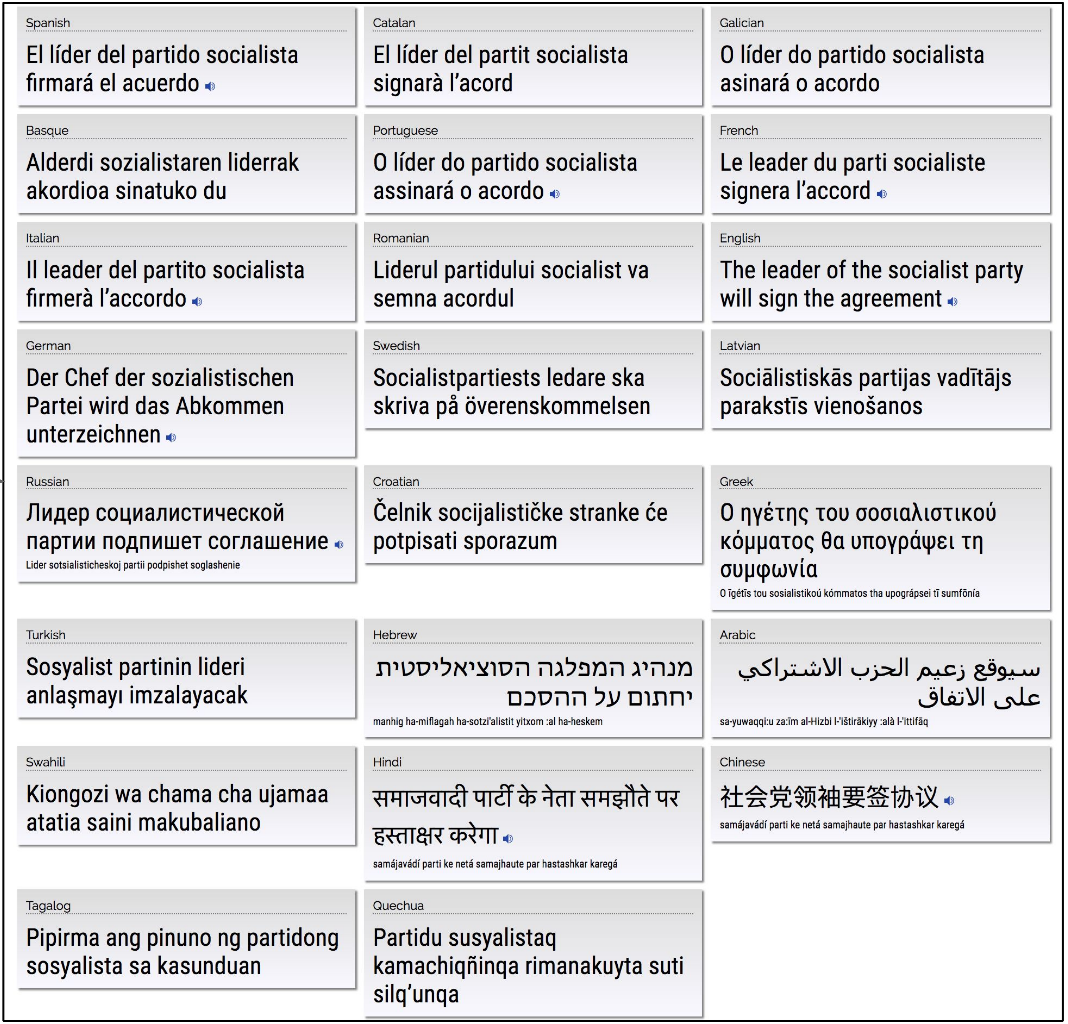
http://www.grammaticalframework.org

Since 1998 (Xerox Research → open source)



Abstract Syntax Tree

linearizations



Spanish
El líder del partido socialista firmará el acuerdo

Catalan
El líder del partit socialista signarà l'acord

Galician
O líder do partido socialista asinará o acordo

Basque
Alderdi sozialistaren liderrak akordioa sinatuko du

Portuguese
O líder do partido socialista assinará o acordo

French
Le leader du parti socialiste signera l'accord

Italian
Il leader del partito socialista firmerà l'accordo

Romanian
Liderul partidului socialist va semna acordul

English
The leader of the socialist party will sign the agreement

German
Der Chef der sozialistischen Partei wird das Abkommen unterzeichnen

Swedish
Socialistpartiests ledare ska skriva på överenskommelsen

Latvian
Sociālistiskās partijas vadītājs parakstīs vienošanos

Russian
Лидер социалистической партии подпишет соглашение
Lider sotsialisticheskoj partii podpishet soglashenie

Croatian
Čelnik socijalističke stranke će potpisati sporazum

Greek
O ηγέτης του σοσιαλιστικού κόμματος θα υπογράψει τη συμφωνία
O īgétīs tou sosialistikoú kómmatos tha upográpsei tī sumfōnía

Turkish
Sosyalist partinin lideri anlaşmayı imzalayacak

Hebrew
מנהיג המפלגה הסוציאליסטית יחתום על ההסכם
manhig ha-miflagah ha-sotzi'alistit yitxom :al ha-heskem

Arabic
سيوقع زعيم الحزب الاشتراكي على الاتفاق
sa-yuwaqqi:u za:īm al-Hizbi l-'ištirākiyy :alà l-'ittifāq

Swahili
Kiongozi wa chama cha ujamaa atatia saini makubaliano

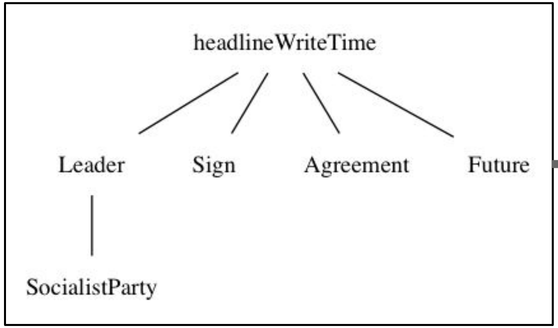
Hindi
समाजवादी पार्टी के नेता समझौते पर हस्ताक्षर करेगा
samájavádí parti ke netá samajhaute par hastashkar karegá

Chinese
社会党领袖要签协议
samájavádí parti ke netá samajhaute par hastashkar karegá

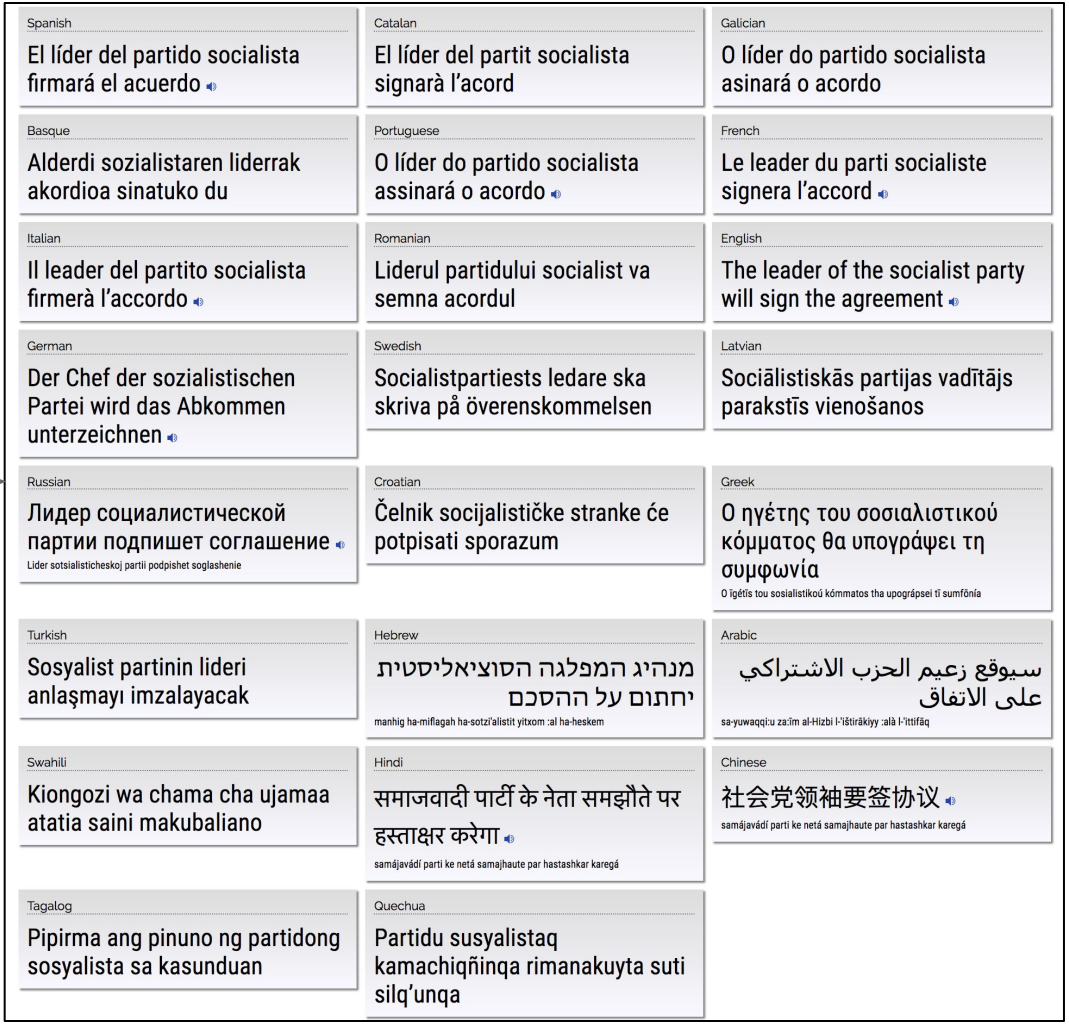
Tagalog
Pipirma ang pinuno ng partidong sosyalista sa kasunduan

Quechua
Partidu susyalistaq kamachiqñinqa rimanakuyta suti silq'unqa

http://www.grammaticalframework.org/demos/multilingual_headlines.html



linearizations
renderers



Spanish
El líder del partido socialista firmará el acuerdo

Catalan
El líder del partit socialista signarà l'acord

Galician
O líder do partido socialista asinará o acordo

Basque
Alderdi sozialistaren liderrak akordioa sinatuko du

Portuguese
O líder do partido socialista assinará o acordo

French
Le leader du parti socialiste signera l'accord

Italian
Il leader del partito socialista firmerà l'accordo

Romanian
Liderul partidului socialist va semna acordul

English
The leader of the socialist party will sign the agreement

German
Der Chef der sozialistischen Partei wird das Abkommen unterzeichnen

Swedish
Socialistpartiests ledare ska skriva på överenskommelsen

Latvian
Sociālistiskās partijas vadītājs parakstīs vienošanos

Russian
Лидер социалистической партии подпишет соглашение
Lider sotsialisticheskoj partii podpishet soglashenie

Croatian
Čelnik socijalističke stranke će potpisati sporazum

Greek
Ο ηγέτης του σοσιαλιστικού κόμματος θα υπογράψει τη συμφωνία
O īgétīs tou sosialistikoú kómmatos tha upográpsei tī sumfōnía

Turkish
Sosyalist partinin lideri anlaşmayı imzalayacak

Hebrew
מנהיג המפלגה הסוציאליסטית יחתום על ההסכם
manhig ha-miflagah ha-sotzi'alistit yitxom :al ha-heskem

Arabic
سيوقع زعيم الحزب الاشتراكي على الاتفاق
sa-yuwaqqi:u za:īm al-Hizbi l-'ištirākiyy :alà l-'ittifāq

Swahili
Kiongozi wa chama cha ujamaa atatia saini makubaliano

Hindi
समाजवादी पार्टी के नेता समझौते पर हस्ताक्षर करेगा
samájavádí parti ke netá samajhaute par hastashkar karegá

Chinese
社会党领袖要签协议
samájavádí parti ke netá samajhaute par hastashkar karegá

Tagalog
Pipirma ang pinuno ng partidong sosyalista sa kasunduan

Quechua
Partidu susyalistaq kamachiqñinqa rimanakuyta suti silq'unqa

Abstract Syntax Tree
constructors

http://www.grammaticalframework.org/demos/multilingual_headlines.html

Support statement from Denny Vrandečić, Wikimedia

“Without Grammatical Framework, Abstract Wikipedia would not exist. Grammatical Framework is a proof by existence that the underlying ideas of Abstract Wikipedia - edit and maintain content in an abstract notation and provide linearizers to create natural language text from the abstract content - are possible and sound."

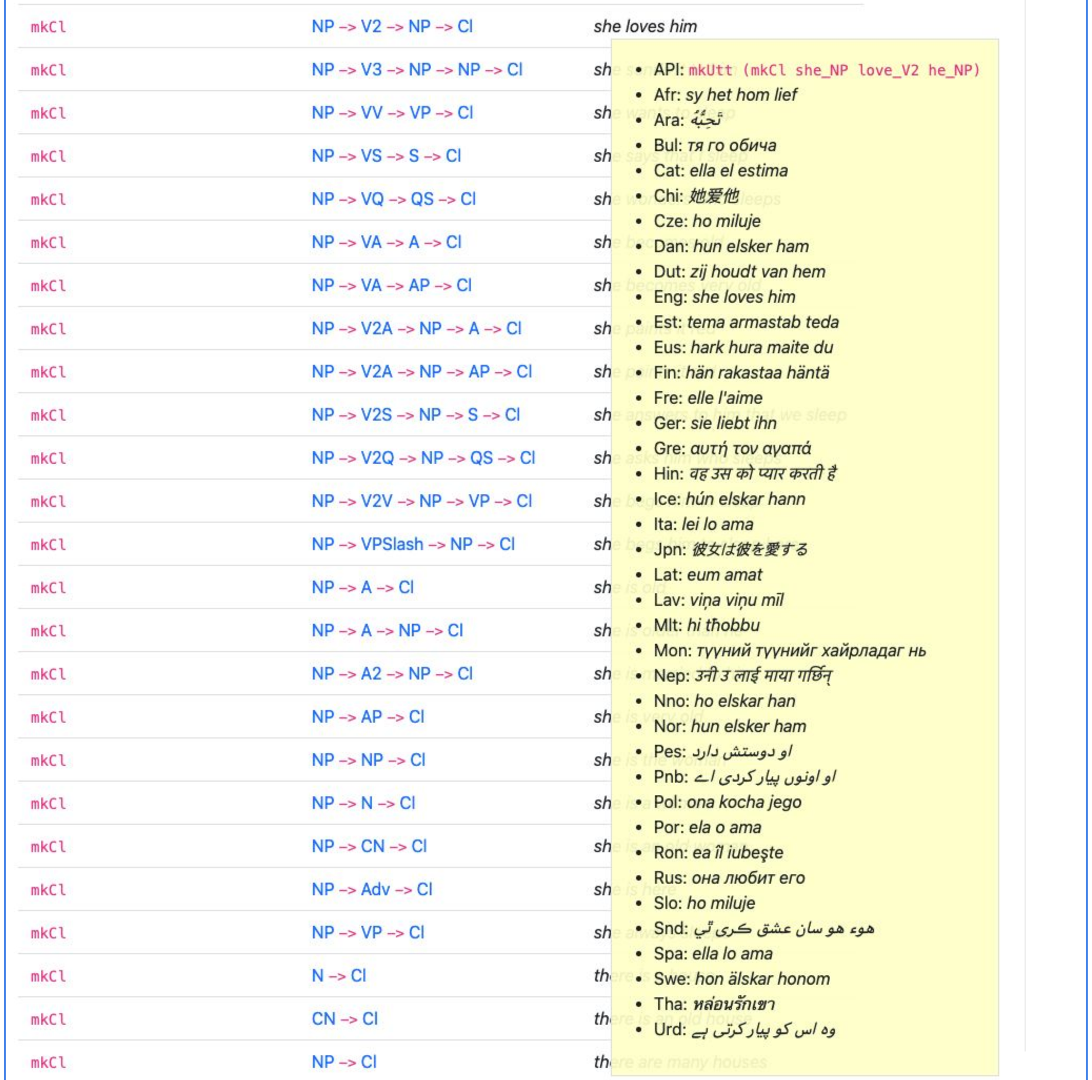
"The experience of the Grammatical Framework researchers and their deep expertise gathered in decades of work will be a unique resource to support us."

(12 August 2021)

RGL = Resource Grammar Library

- syntax API

http://www.grammaticalframework.org/lib/doc/synopsis/



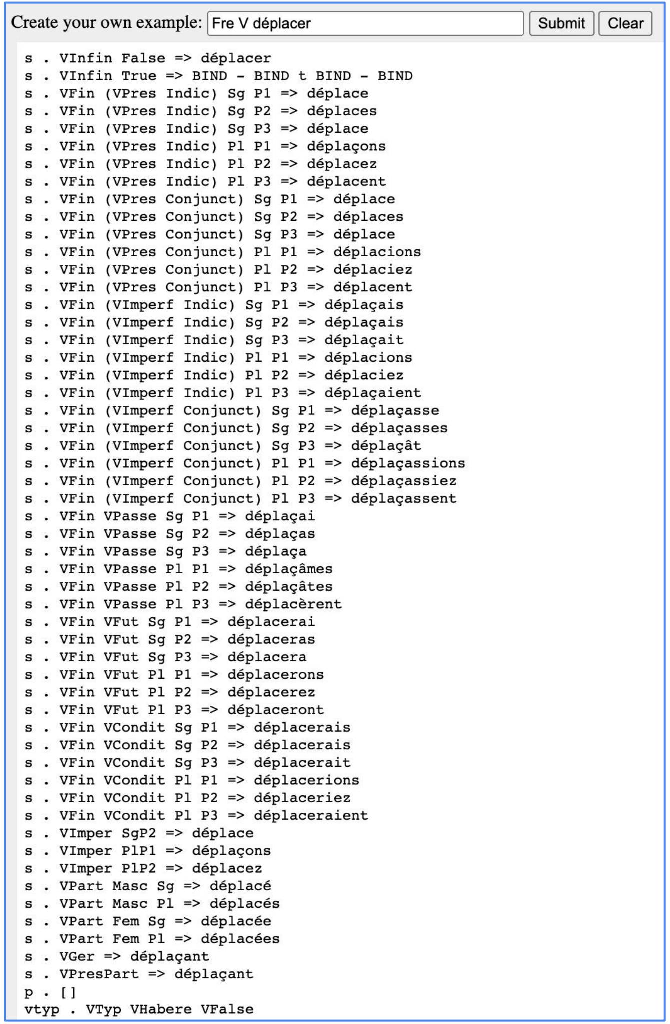
mkCl	NP -> V2 -> NP -> Cl	she loves him
mkCl	NP -> V3 -> NP -> NP -> Cl	sh
mkCl	NP -> VV -> VP -> Cl	sh
mkCl	NP -> VS -> S -> Cl	sh
mkCl	NP -> VQ -> QS -> Cl	sh
mkCl	NP -> VA -> A -> Cl	sh
mkCl	NP -> VA -> AP -> Cl	sh
mkCl	NP -> V2A -> NP -> A -> Cl	sh
mkCl	NP -> V2A -> NP -> AP -> Cl	sh
mkCl	NP -> V2S -> NP -> S -> Cl	sh
mkCl	NP -> V2Q -> NP -> QS -> Cl	sh
mkCl	NP -> V2V -> NP -> VP -> Cl	sh
mkCl	NP -> VPSlash -> NP -> Cl	sh
mkCl	NP -> A -> Cl	sh
mkCl	NP -> A -> NP -> Cl	sh
mkCl	NP -> A2 -> NP -> Cl	sh
mkCl	NP -> AP -> Cl	sh
mkCl	NP -> NP -> Cl	sh
mkCl	NP -> N -> Cl	sh
mkCl	NP -> CN -> Cl	sh
mkCl	NP -> Adv -> Cl	sh
mkCl	NP -> VP -> Cl	sh
mkCl	N -> Cl	th
mkCl	CN -> Cl	th
mkCl	NP -> Cl	th

- API: `mkUtt (mkCl she_NP love_V2 he_NP)`
- Afr: *sy het hom lief*
- Ara: *تَحِبُّهُ*
- Bul: *тя го обича*
- Cat: *ella el estima*
- Chi: *她爱他*
- Cze: *ho miluje*
- Dan: *hun elsker ham*
- Dut: *zij houdt van hem*
- Eng: *she loves him*
- Est: *tema armastab teda*
- Eus: *hark hura maite du*
- Fin: *hän rakastaa häntä*
- Fre: *elle l'aime*
- Ger: *sie liebt ihn*
- Gre: *αυτή τον αγαπά*
- Hin: *वह उस को प्यार करती है*
- Ice: *hún elskar hann*
- Ita: *lei lo ama*
- Jpn: *彼女は彼を愛する*
- Lat: *eum amat*
- Lav: *viņa viņu mīl*
- Mlt: *hi tħobbu*
- Mon: *түүний түүнийг хайрладаг нь*
- Nep: *उनी उ लाई माया गर्छिन्*
- Nno: *ho elskar han*
- Nor: *hun elsker ham*
- Pes: *او دوستش دارد*
- Pnb: *او اونوں پیار کردی اے*
- Pol: *ona kocha jego*
- Por: *ela o ama*
- Ron: *ea îl iubeşte*
- Rus: *она любит его*
- Slo: *ho miluje*
- Snd: *هوء هو سان عشق ڪري ٿي*
- Spa: *ella lo ama*
- Swe: *hon älskar honom*
- Tha: *หล่อนรักเขา*
- Urd: *وہ اس کو پیار کرتی ہے*

RGL = Resource Grammar Library

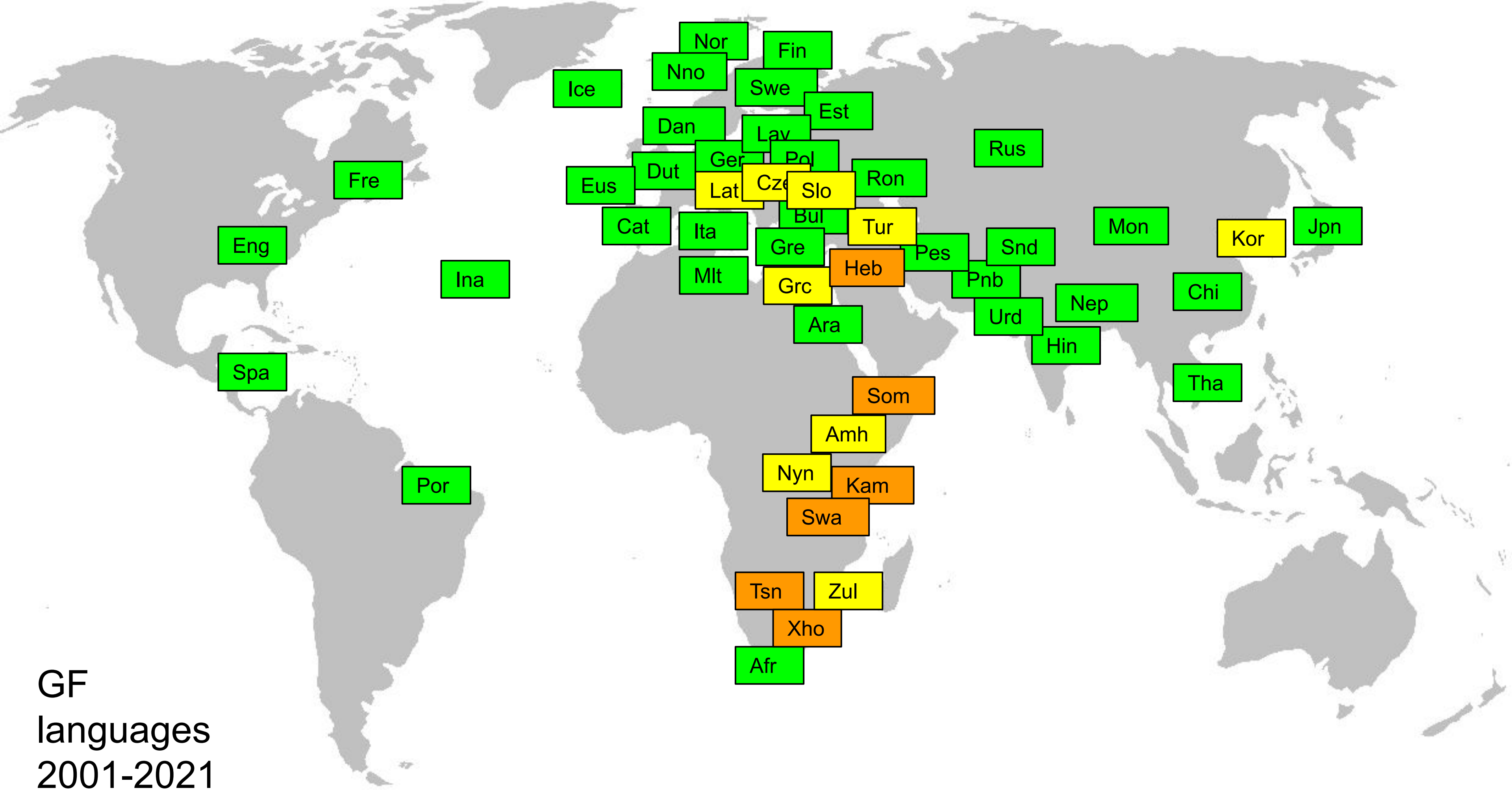
- smart paradigms

http://cloud.grammaticalframework.org/gfmorpho/



Create your own example: Fre V déplacer

```
s . VInfin False => déplacer
s . VInfin True => BIND - BIND t BIND - BIND
s . VFin (VPres Indic) Sg P1 => déplace
s . VFin (VPres Indic) Sg P2 => déplaces
s . VFin (VPres Indic) Sg P3 => déplace
s . VFin (VPres Indic) Pl P1 => déplaçons
s . VFin (VPres Indic) Pl P2 => déplacez
s . VFin (VPres Indic) Pl P3 => déplacent
s . VFin (VPres Conjunct) Sg P1 => déplace
s . VFin (VPres Conjunct) Sg P2 => déplaces
s . VFin (VPres Conjunct) Sg P3 => déplace
s . VFin (VPres Conjunct) Pl P1 => déplacions
s . VFin (VPres Conjunct) Pl P2 => déplaciez
s . VFin (VPres Conjunct) Pl P3 => déplacent
s . VFin (VImperf Indic) Sg P1 => déplaçais
s . VFin (VImperf Indic) Sg P2 => déplaçais
s . VFin (VImperf Indic) Sg P3 => déplaçait
s . VFin (VImperf Indic) Pl P1 => déplacions
s . VFin (VImperf Indic) Pl P2 => déplaciez
s . VFin (VImperf Indic) Pl P3 => déplaçaient
s . VFin (VImperf Conjunct) Sg P1 => déplaçasse
s . VFin (VImperf Conjunct) Sg P2 => déplaçasses
s . VFin (VImperf Conjunct) Sg P3 => déplaçât
s . VFin (VImperf Conjunct) Pl P1 => déplaçassions
s . VFin (VImperf Conjunct) Pl P2 => déplaçassiez
s . VFin (VImperf Conjunct) Pl P3 => déplaçassent
s . VFin VPasse Sg P1 => déplaçai
s . VFin VPasse Sg P2 => déplaças
s . VFin VPasse Sg P3 => déplaça
s . VFin VPasse Pl P1 => déplaçâmes
s . VFin VPasse Pl P2 => déplaçâtes
s . VFin VPasse Pl P3 => déplacèrent
s . VFin VFut Sg P1 => déplacerai
s . VFin VFut Sg P2 => déplaceras
s . VFin VFut Sg P3 => déplacera
s . VFin VFut Pl P1 => déplacerons
s . VFin VFut Pl P2 => déplacerez
s . VFin VFut Pl P3 => déplaceront
s . VFin VCondit Sg P1 => déplacerais
s . VFin VCondit Sg P2 => déplacerais
s . VFin VCondit Sg P3 => déplacerait
s . VFin VCondit Pl P1 => déplacerions
s . VFin VCondit Pl P2 => déplaceriez
s . VFin VCondit Pl P3 => déplaceraient
s . VImper SgP2 => déplace
s . VImper PlP1 => déplaçons
s . VImper PlP2 => déplacez
s . VPart Masc Sg => déplacé
s . VPart Masc Pl => déplacés
s . VPart Fem Sg => déplacée
s . VPart Fem Pl => déplacées
s . VGer => déplaçant
s . VPresPart => déplaçant
p . []
vtyp . VTyp VHabere VFalse
```

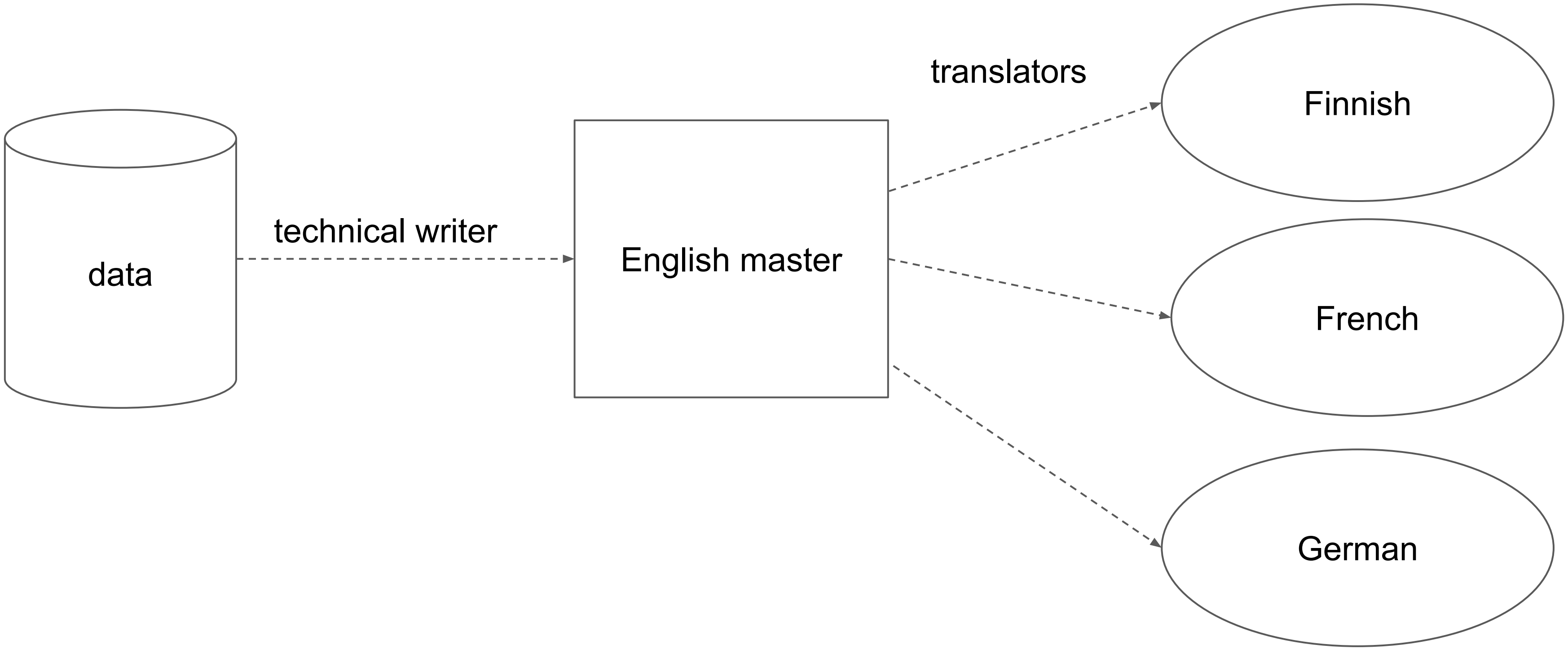


GF
languages
2001-2021

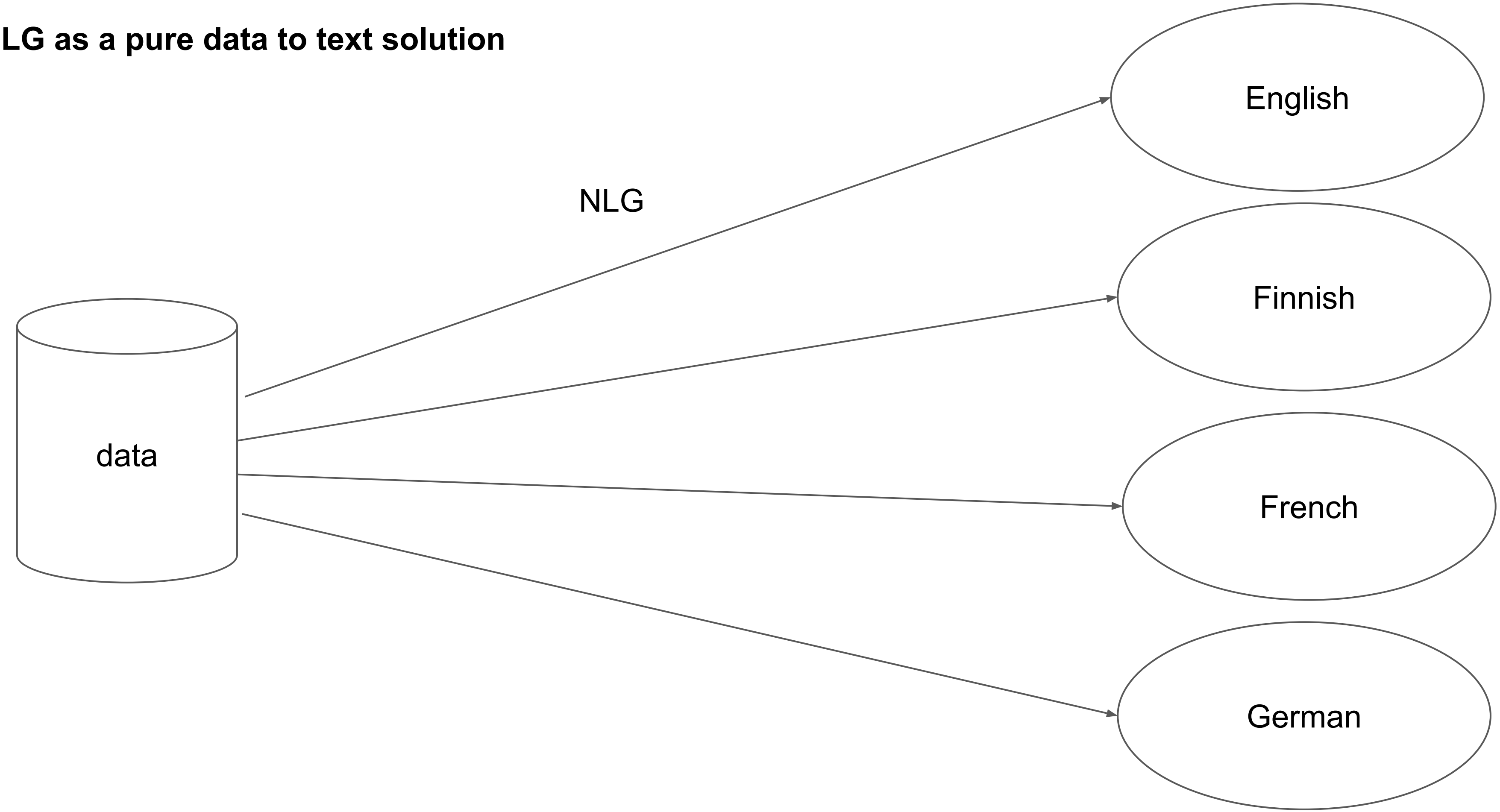
https://commons.wikimedia.org/wiki/Maps_of_the_world#/media/File:BlankMap-World-noborders.png

NLG in GF: baseline and extensions

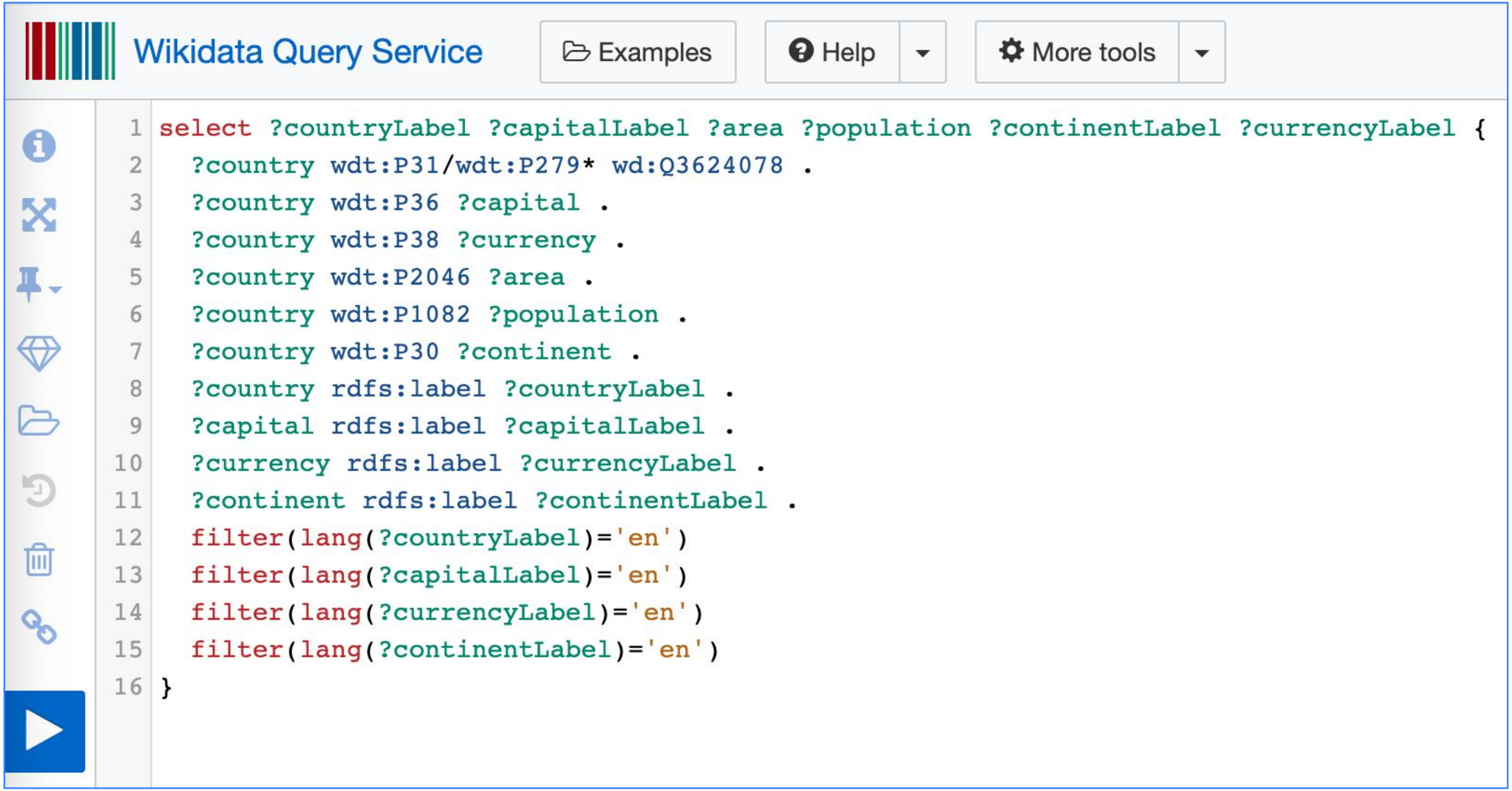
traditional workflow for multilingual documentation



NLG as a pure data to text solution

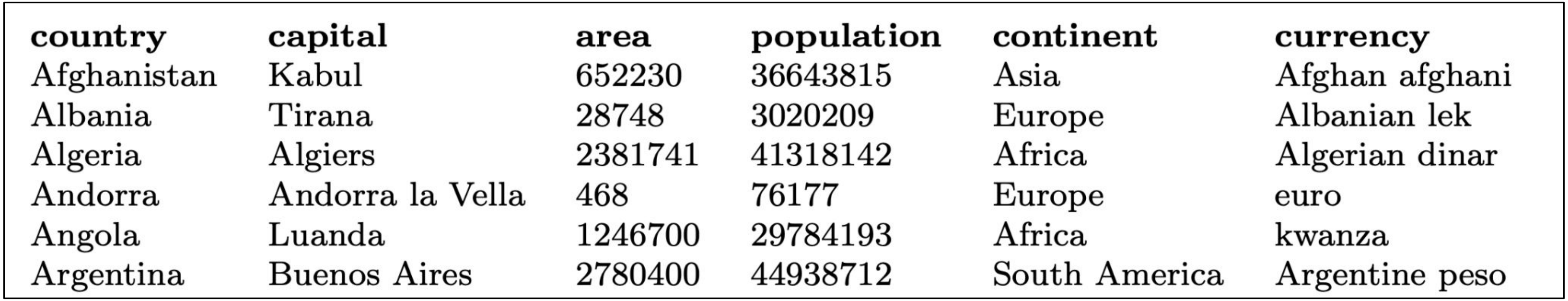


An experiment, starting from Wikidata



```
select ?countryLabel ?capitalLabel ?area ?population ?continentLabel ?currencyLabel {
  ?country wdt:P31/wdt:P279* wd:Q3624078 .
  ?country wdt:P36 ?capital .
  ?country wdt:P38 ?currency .
  ?country wdt:P2046 ?area .
  ?country wdt:P1082 ?population .
  ?country wdt:P30 ?continent .
  ?country rdfs:label ?countryLabel .
  ?capital rdfs:label ?capitalLabel .
  ?currency rdfs:label ?currencyLabel .
  ?continent rdfs:label ?continentLabel .
  filter(lang(?countryLabel)='en')
  filter(lang(?capitalLabel)='en')
  filter(lang(?currencyLabel)='en')
  filter(lang(?continentLabel)='en')
}
```

https://w.wiki/3fV$



country	capital	area	population	continent	currency
Afghanistan	Kabul	652230	36643815	Asia	Afghan afghani
Albania	Tirana	28748	3020209	Europe	Albanian lek
Algeria	Algiers	2381741	41318142	Africa	Algerian dinar
Andorra	Andorra la Vella	468	76177	Europe	euro
Angola	Luanda	1246700	29784193	Africa	kwanza
Argentina	Buenos Aires	2780400	44938712	South America	Argentine peso

Bottom-up approach

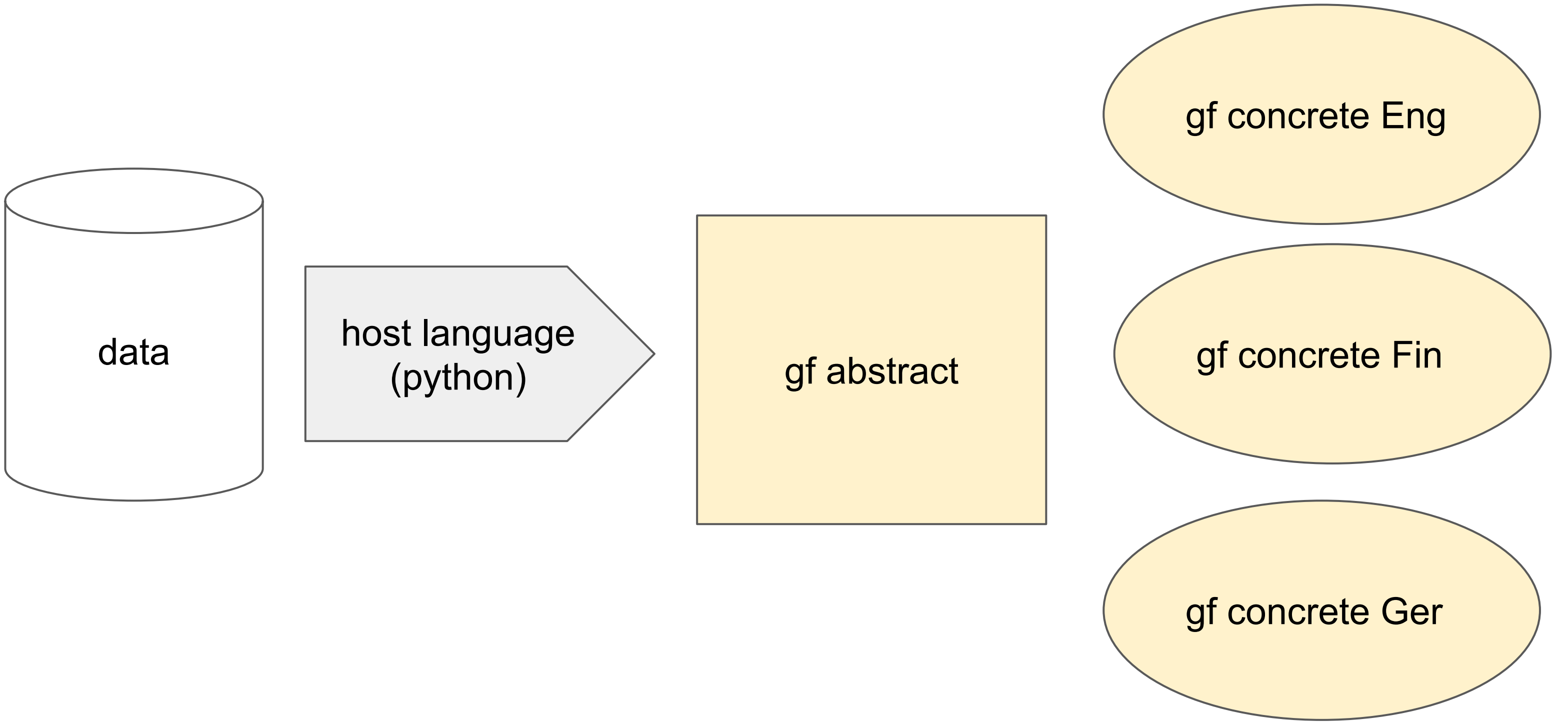
Stage 1: templates for atomic country facts

Stage 2: grammars for atomic facts

Stage 3: text planning

Stage 4: content planning

System architecture



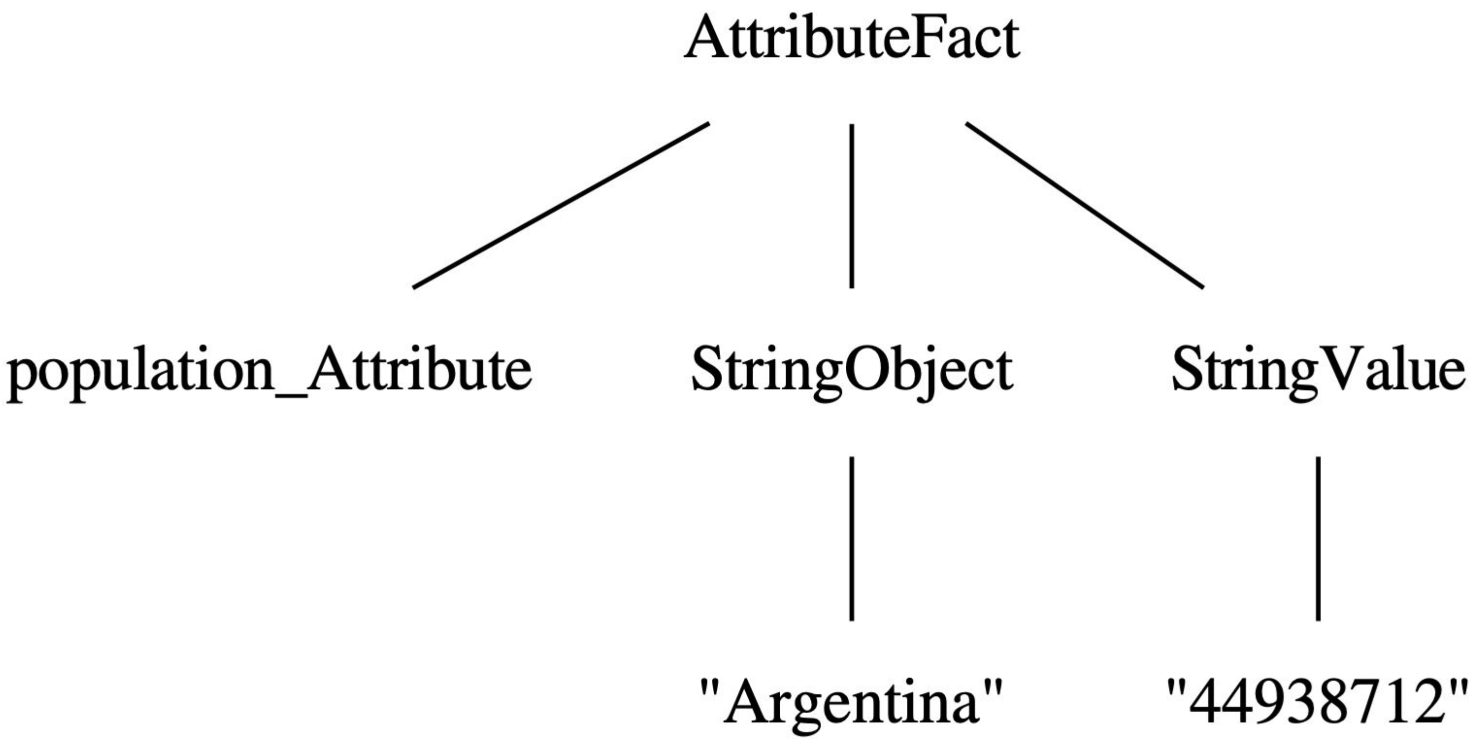
Stage 1:

Template-based generation of atomic facts

the capital of Argentina is Buenos Aires
the area of Argentina is 2780400
the population of Argentina is 44938712
the continent of Argentina is South America
the currency of Argentina is Argentine peso

```
abstract Facts = {

cat
  Fact ;
  Object ;
  Attribute ;
  Value ;
fun
  AttributeFact : Attribute -> Object -> Value -> Fact ;

  capital_Attribute : Attribute ;
  area_Attribute : Attribute ;
  population_Attribute : Attribute ;
  continent_Attribute : Attribute ;
  currency_Attribute : Attribute ;

  StringObject : String -> Object ;
  StringValue : String -> Value ;
}
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts1/Facts.gf


AttributeFact
population_Attribute
StringObject
StringValue
"Argentina"
"44938712"

```
concrete FactsEng of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;

lin
  AttributeFact attr obj val =
    "the" ++ attr ++ "of" ++ obj ++ "is" ++ val ;

  capital_Attribute = "capital" ;
  area_Attribute = "area" ;
  population_Attribute = "population" ;
  continent_Attribute = "continent" ;
  currency_Attribute = "currency" ;

  StringObject str = str.s ;
  StringValue str = str.s ;
}
```

```
concrete FactsGer of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;

lin
  AttributeFact attr obj val =
    attr ++ "von" ++ obj ++ "ist" ++ val ;

  capital_Attribute = "die Hauptstadt" ;
  area_Attribute = "die Fläche" ;
  population_Attribute = "die Einwohnerzahl" ;
  continent_Attribute = "der Kontinent" ;
  currency_Attribute = "die Währung" ;

  StringObject str = str.s ;
  StringValue str = str.s ;
}
```

die Hauptstadt von Argentina ist Buenos Aires
die Fläche von Argentina ist 2780400
die Einwohnerzahl von Argentina ist 44938712
der Kontinent von Argentina ist South America
die Währung von Argentina ist Argentine peso

```
concrete FactsFin of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;

lin
  AttributeFact attr obj val =
    "maan" ++ obj ++ attr ++ "on" ++ val ;

  capital_Attribute = "pääkaupunki" ;
  area_Attribute = "pinta-ala" ;
  population_Attribute = "asukasluku" ;
  continent_Attribute = "maanosa" ;
  currency_Attribute = "valuutta" ;

  StringObject str = str.s ;
  StringValue str = str.s ;
}
```

maan Argentina pääkaupunki on Buenos Aires
maan Argentina pinta-ala on 2780400
maan Argentina asukasluku on 44938712
maan Argentina maanosa on South America
maan Argentina valuutta on Argentine peso

Portable Grammar Format, making GF accessible from Python, Java, C, Haskell,...

```
import pgf

def country_facts(c):
  object = pgf.Expr('StringObject',[string_expr(c.country)])
  return [
    pgf.Expr('AttributeFact',
           [pgf.Expr(attr,[]),object,pgf.Expr('StringValue',[string_expr(val)])])

      for (attr,val) in [
        ('capital_Attribute', c.capital),
        ('area_Attribute', c.area),
        ('population_Attribute', c.population),
        ('continent_Attribute', c.continent),
        ('currency_Attribute', c.currency)
        ]
    ]
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts1/facts.py

```
def main():
    gr = pgf.readPGF(pgf_file)
    countries = get_countries(country_file)
    langs = list(gr.languages.values())
    for lang in langs:
        text = []
        for c in countries:
            for t in country_facts(c):
                text.append(lang.linearize(t))
        print('\n'.join(text))
```

How GF helps, #1

Using a multilingual GF grammar and its abstract syntax,

we can define NLG simultaneously for any number of languages.

maan Argentina pääkaupunki on Buenos Aires
maan Argentina pinta-ala on 2780400
maan Argentina asukasluku on 44938712
maan Argentina maanosa on South America
maan Argentina valuutta on Argentine peso

Stage 2:

Grammar-based generation of atomic facts

the population of United States of America is 331449281

→ (**grammar for names**)

the population of the United States is 331449281

→ (**attribute-specific rendering**)

the United States has 331449281 inhabitants die Einwohnerzahl von United States of America ist 331449281

→ (**grammar for names**)

die Einwohnerzahl von den Vereinigten Staaten ist 331449281

→ (**attribute-specific rendering**)

die Vereinigten Staaten haben 331449281 Einwohner maan United States of America asukasluku 331449281

→ (**grammar for names**)

Yhdysvaltain asukasluku on 331449281

→ (**attribute-specific rendering**)

Yhdysvalloissa on 331449281 asukasta

```
concrete FactsEng of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;
lin
  AttributeFact attr obj val =
    "the" ++ attr ++
       "of" ++ obj ++ "is" ++ val ;
```

```
concrete FactsEng of Facts =
                  open SyntaxEng in {
lincat
  Fact = Cl ;
  Object = NP ;
  Attribute = CN ;
  Value = NP ;
lin
  AttributeFact attr obj val =
    mkCl (mkNP the_Det (mkCN attr
       (mkAdv possess_Prep obj))) val ;
```

```
concrete FactsEng of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;
lin
  AttributeFact attr obj val =
    "the" ++ attr ++
       "of" ++ obj ++ "is" ++ val ;
```

```
concrete FactsEng of Facts =
                  open SyntaxEng in {
lincat
  Fact = Cl ;
  Object = NP ;
  Attribute = CN ;
  Value = NP ;
lin
  AttributeFact attr obj val =
    mkCl (mkNP the_Det (mkCN attr
       (mkAdv possess_Prep obj))) val ;
```

```
concrete FactsGer of Facts = {

  AttributeFact attr obj val =
    attr ++
     "von" ++ obj ++ "ist" ++ val ;
```

```
concrete FactsGer of Facts =
                   open SyntaxGer in {
  AttributeFact attr obj val =
    mkCl (mkNP the_Det (mkCN attr
       (mkAdv possess_Prep obj))) val ;
```

```
concrete FactsEng of Facts = {

lincat
  Fact = Str ;
  Object = Str ;
  Attribute = Str ;
  Value = Str ;
lin
  AttributeFact attr obj val =
    "the" ++ attr ++
       "of" ++ obj ++ "is" ++ val ;
```

```
concrete FactsEng of Facts =
                  open SyntaxEng in {
lincat
  Fact = Cl ;
  Object = NP ;
  Attribute = CN ;
  Value = NP ;
lin
  AttributeFact attr obj val =
    mkCl (mkNP the_Det (mkCN attr
       (mkAdv possess_Prep obj))) val ;
```

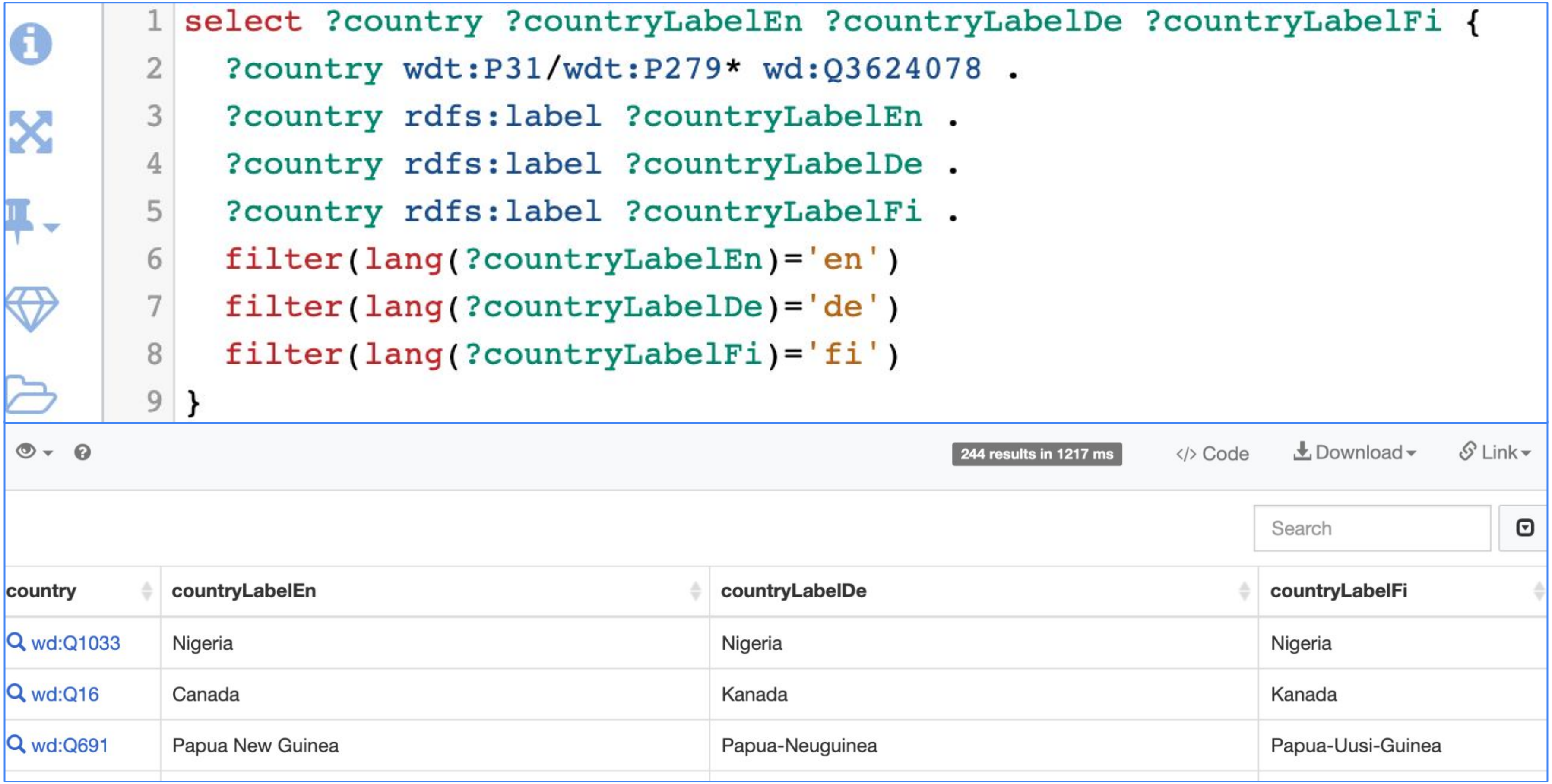
```
concrete FactsGer of Facts = {

  AttributeFact attr obj val =
    attr ++
     "von" ++ obj ++ "ist" ++ val ;
```

```
concrete FactsGer of Facts =
                   open SyntaxGer in {
  AttributeFact attr obj val =
    mkCl (mkNP the_Det (mkCN attr
       (mkAdv possess_Prep obj))) val ;
```

```
concrete FactsFin of Facts = {
  AttributeFact attr obj val =
    "maan" ++ obj ++
        attr ++ "on" ++ val ;
```

```
concrete FactsFin of Facts =
             open SyntaxFin, (E=ExtendFin) in {
  AttributeFact attr obj val =
    mkCl (mkNP (E.GenNP obj) attr) val ;
```

Lexicon from WikiData labels



```
select ?country ?countryLabelEn ?countryLabelDe ?countryLabelFi {
  ?country wdt:P31/wdt:P279* wd:Q3624078 .
  ?country rdfs:label ?countryLabelEn .
  ?country rdfs:label ?countryLabelDe .
  ?country rdfs:label ?countryLabelFi .
  filter(lang(?countryLabelEn)='en')
  filter(lang(?countryLabelDe)='de')
  filter(lang(?countryLabelFi)='fi')
}
```

country	countryLabelEn	countryLabelDe	countryLabelFi
wd:Q1033	Nigeria	Nigeria	Nigeria
wd:Q16	Canada	Kanada	Kanada
wd:Q691	Papua New Guinea	Papua-Neuguinea	Papua-Uusi-Guinea

https://w.wiki/3mWG

```
aarne$ python3 extract_names.py


fun United_States_of_America_CName : CName ;
lin United_States_of_America_CName = mkCName "Vereinigte Staaten" ;
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts2/extract_names.py

```
aarne$ python3 extract_names.py


fun United_States_of_America_CName : CName ;
lin United_States_of_America_CName = mkCName "Vereinigte Staaten" ;
```

```
lin United_States_of_America_CName =
  mkCName (mkNP thePl_Det (mkCN (mkA "Vereinigt") (mkN "Staat" "Staaten")))
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts2/extract_names.py

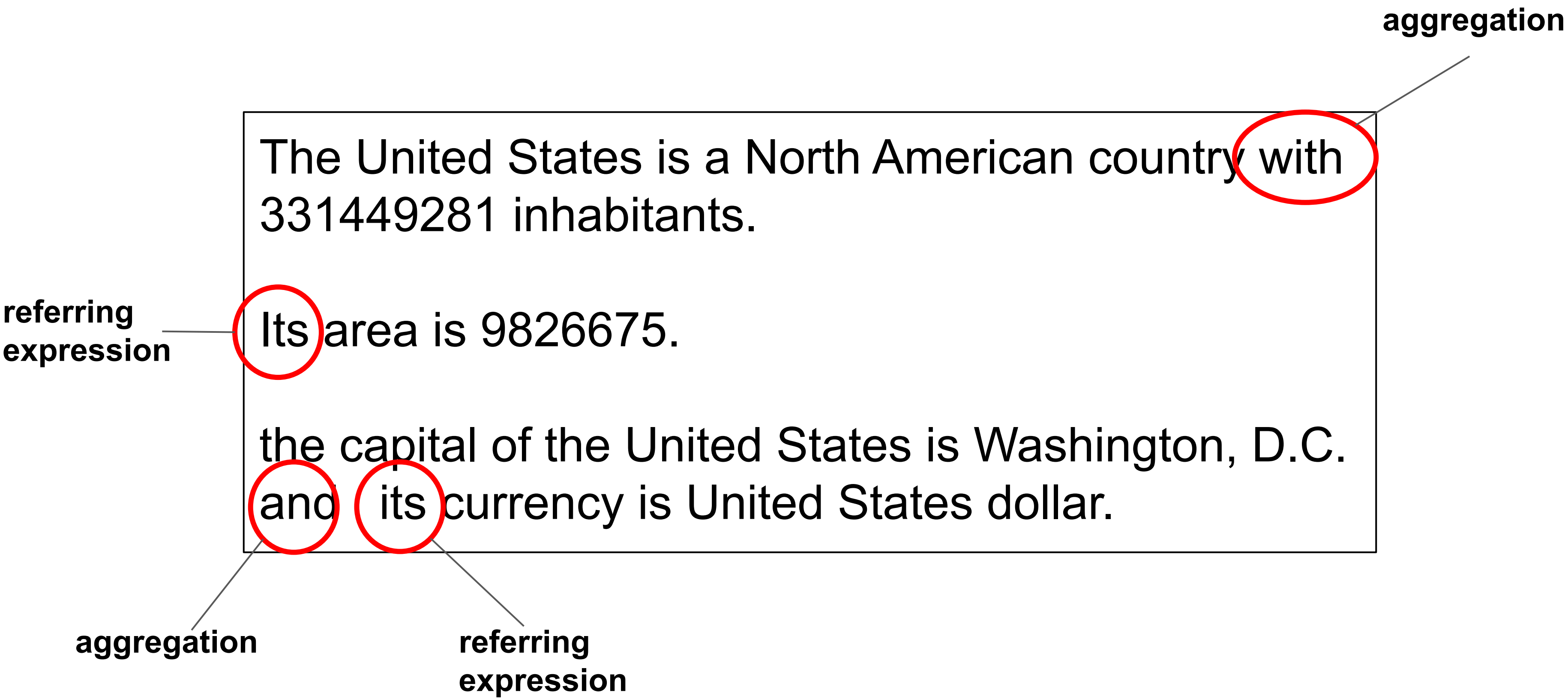
How GF helps, #2

Using GF-RGL and lexical data (such as Wikidata "labels"),

we can build syntax rules and a lexicon for a new language almost automatically.

Stage 3:

Building a fluent text



The United States is a North American country with 331449281 inhabitants.

Its area is 9826675.

the capital of the United States is Washington, D.C. and its currency is United States dollar.

NLG function v1: abstract syntax via an **embedded GF grammar**

```
def country_texts_embedded(factsys,data):
...
        doc = G.OneSentenceDoc(
                G.FactSentence(
                    G.KindFact(G.NameObject(countr),
                        G.ModifierKind(G.PropertyKind(G.cdProperty(cont),G.country_Kind),
                            G.NumericKindModifier(G.IntNumeric(pop),G.inhabitant_Kind)))))
        doc = G.AddSentenceDoc(doc,
                   G.FactSentence(G.AttributeFact(G.area_Attribute, G.PronObject(countr),
                                                      G.NumericValue(G.IntNumeric(are)))))
        doc = G.AddSentenceDoc(doc,
                 G.ConjSentence(
                   G.FactSentence(G.AttributeFact(G.capital_Attribute, G.NameObject(countr), G.NameValue(cap))),
                   G.FactSentence(G.AttributeFact(G.currency_Attribute, G.PronObject(countr), G.NameValue(curr)))))
...
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts3/country_facts.py

NLG function v2: abstract syntax via GF's **parser** on text templates

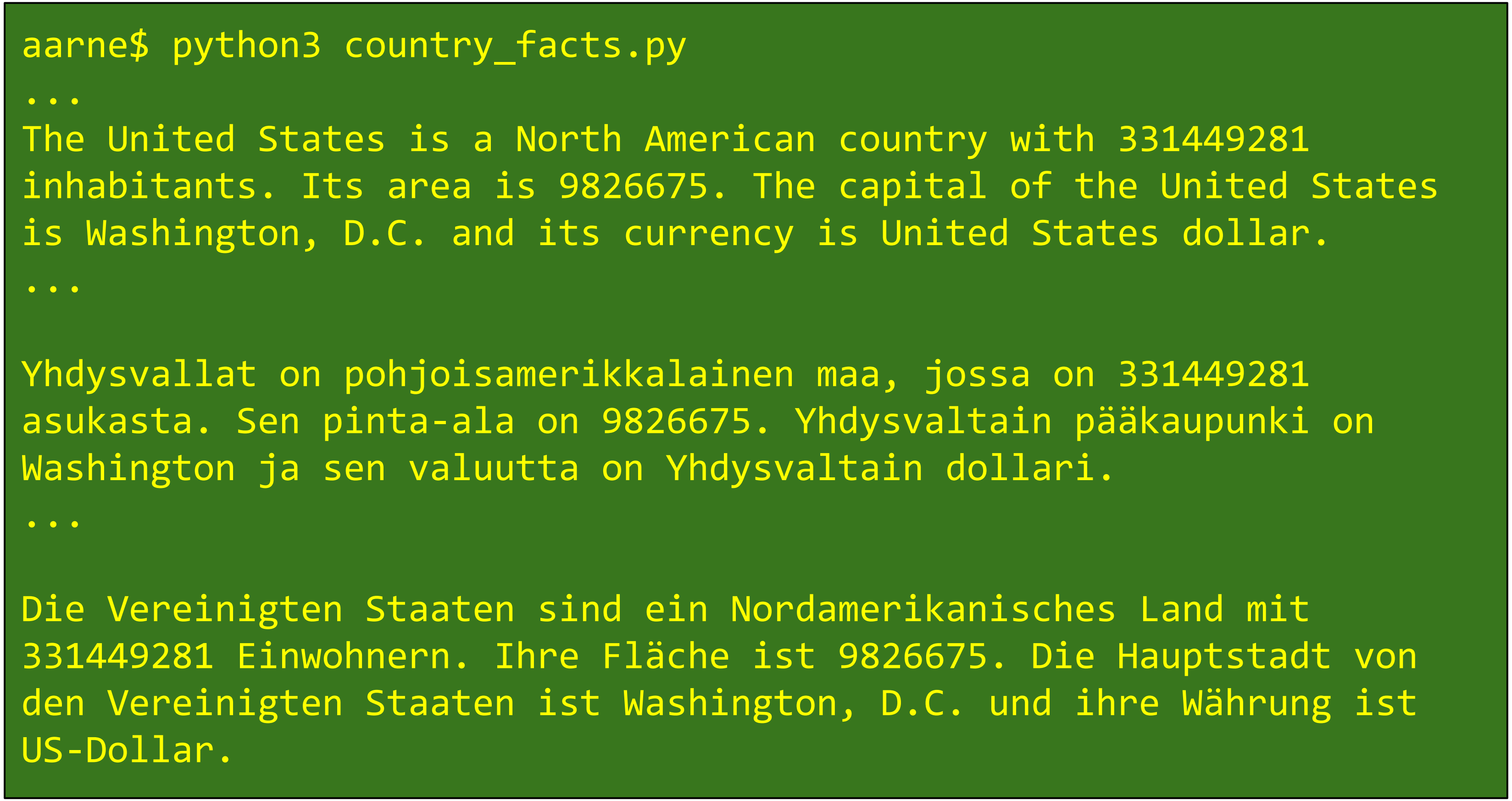
```
def country_texts_parsed(factsys,data):
  ...
  doc = factsys.str2exp("Doc",
          ("{} is a {} country with {} inhabitants. "
           "its area is {} . "
           "the capital of {} is {} and its currency is {}.").
                       format(countr,cont,pop,are,countr,cap,curr))
  ...
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts3/country_facts.py

How GF helps, #3

Using the parser of GF,

we can write complex grammar rules that look like templates.



```
aarne$ python3 country_facts.py
...
The United States is a North American country with 331449281
inhabitants. Its area is 9826675. The capital of the United States
is Washington, D.C. and its currency is United States dollar.
...

Yhdysvallat on pohjoisamerikkalainen maa, jossa on 331449281
asukasta. Sen pinta-ala on 9826675. Yhdysvaltain pääkaupunki on
Washington ja sen valuutta on Yhdysvaltain dollari.
...

Die Vereinigten Staaten sind ein Nordamerikanisches Land mit
331449281 Einwohnern. Ihre Fläche ist 9826675. Die Hauptstadt von
den Vereinigten Staaten ist Washington, D.C. und ihre Währung ist
US-Dollar.
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts3/country_facts.py

```
concrete FactsEng of Facts = open
  SyntaxEng,
  SymbolicEng,
  GrammarEng,
  Prelude
in {
lincat
  Doc = Text ;
  Sentence = S ;
  Fact = Cl ;
  Object = {np : NP ; pron : Pron ; isPron : Bool} ;
  Property = AP ;
  Attribute = CN ;
  Modifier = {adv : Adv ; rs : RS ; isAdv : Bool} ;
  Kind = CN ;
  Value = NP ;
  Name = NP ;
  Numeric = Card ;
lin
  OneSentenceDoc sent = mkText sent ;
  AddSentenceDoc doc sent = mkText doc (mkText sent) ;
  ConjSentence a b = mkS and_Conj a b ;
  FactSentence fact = mkS presentTense positivePol fact ;
  KindFact obj kind = mkCl obj.np (mkNP a_Det kind) ; --- sind ein Land
  PropertyFact obj prop = mkCl obj.np prop ;
  AttributeFact attr obj val = case obj.isPron of {
    True => mkCl (mkNP (mkDet obj.pron) attr) val ;
    _ => mkCl (mkNP the_Det (mkCN attr (mkAdv possess_Prep obj.np))) val
    } ;
  PropertyKind prop kind = mkCN prop kind ;
  ModifierKind kind mod = case mod.isAdv of {
    False => mkCN kind mod.rs ;
    True => mkCN kind mod.adv
    } ;
  NumericKindModifier num kind = mkModifier (mkAdv with_Prep (mkNP num kind)) ;
```

```
concrete FactsGer of Facts = open
  SyntaxGer,
  SymbolicGer,
  GrammarGer,
  Prelude
in {
lincat
  Doc = Text ;
  Sentence = S ;
  Fact = Cl ;
  Object = {np : NP ; pron : Pron ; isPron : Bool} ;
  Property = AP ;
  Attribute = CN ;
  Modifier = {adv : Adv ; rs : RS ; isAdv : Bool} ;
  Kind = CN ;
  Value = NP ;
  Name = NP ;
  Numeric = Card ;
lin
  OneSentenceDoc sent = mkText sent ;
  AddSentenceDoc doc sent = mkText doc (mkText sent) ;
  ConjSentence a b = mkS and_Conj a b ;
  FactSentence fact = mkS presentTense positivePol fact ;
  KindFact obj kind = mkCl obj.np (mkNP a_Det kind) ; --- sind ein Land
  PropertyFact obj prop = mkCl obj.np prop ;
  AttributeFact attr obj val = case obj.isPron of {
    True => mkCl (mkNP (mkDet obj.pron) attr) val ;
    _ => mkCl (mkNP the_Det (mkCN attr (mkAdv possess_Prep obj.np))) val
    } ;
  PropertyKind prop kind = mkCN prop kind ;
  ModifierKind kind mod = case mod.isAdv of {
    False => mkCN kind mod.rs ;
    True => mkCN kind mod.adv
    } ;
  NumericKindModifier num kind = mkModifier (mkAdv with_Prep (mkNP num kind)) ;
```

```
concrete FactsEng of Facts = open
  SyntaxEng,
  SymbolicEng,
  GrammarEng,
  Prelude
in {
lincat
  Doc = Text ;
  Sentence = S ;
  Fact = Cl ;
  Object = {np : NP ; pron : Pron ; isPron : Bool} ;
  Property = AP ;
  Attribute = CN ;
  Modifier = {adv : Adv ; rs : RS ; isAdv : Bool} ;
  Kind = CN ;
  Value = NP ;
  Name = NP ;
  Numeric = Card ;
lin
  OneSentenceDoc sent = mkText sent ;
  AddSentenceDoc doc sent = mkText doc (mkText sent) ;
  ConjSentence a b = mkS and_Conj a b ;
  FactSentence fact = mkS presentTense positivePol fact ;
  KindFact obj kind = mkCl obj.np (mkNP a_Det kind) ; --- sind ein Land
  PropertyFact obj prop = mkCl obj.np prop ;
  AttributeFact attr obj val = case obj.isPron of {
    True => mkCl (mkNP (mkDet obj.pron) attr) val ;
    _ => mkCl (mkNP the_Det (mkCN attr (mkAdv possess_Prep obj.np))) val
    } ;
  PropertyKind prop kind = mkCN prop kind ;
  ModifierKind kind mod = case mod.isAdv of {
    False => mkCN kind mod.rs ;
    True => mkCN kind mod.adv
    } ;
  NumericKindModifier num kind = mkModifier (mkAdv with_Prep (mkNP num kind)) ;
```

```
concrete FactsGer of Facts = open
  SyntaxGer,
  SymbolicGer,
  GrammarGer,
  Prelude
in {
lincat
  Doc = Text ;
  Sentence = S ;
  Fact = Cl ;
  Object = {np : NP ; pron : Pron ; isPron : Bool} ;
  Property = AP ;
  Attribute = CN ;
  Modifier = {adv : Adv ; rs : RS ; isAdv : Bool} ;
  Kind = CN ;
  Value = NP ;
  Name = NP ;
  Numeric = Card ;
lin
  OneSentenceDoc sent = mkText sent ;
  AddSentenceDoc doc sent = mkText doc (mkText sent) ;
  ConjSentence a b = mkS and_Conj a b ;
  FactSentence fact = mkS presentTense positivePol fact ;
  KindFact obj kind = mkCl obj.np (mkNP a_Det kind) ; --- sind ein Land
  PropertyFact obj prop = mkCl obj.np prop ;
  AttributeFact attr obj val = case obj.isPron of {
    True => mkCl (mkNP (mkDet obj.pron) attr) val ;
    _ => mkCl (mkNP the_Det (mkCN attr (mkAdv possess_Prep obj.np))) val
    } ;
  PropertyKind prop kind = mkCN prop kind ;
  ModifierKind kind mod = case mod.isAdv of {
    False => mkCN kind mod.rs ;
    True => mkCN kind mod.adv
    } ;
  NumericKindModifier num kind = mkModifier (mkAdv with_Prep (mkNP num kind)) ;
```

```
incomplete concrete FactsFunctor of Facts = open
  Syntax,
  Symbolic,
  Grammar,
  Prelude
in {
lincat
  Doc = Text ;
  Sentence = S ;
  Fact = Cl ;
  Object = {np : NP ; pron : Pron ; isPron : Bool} ;
  Property = AP ;
  Attribute = CN ;
  Modifier = {adv : Adv ; rs : RS ; isAdv : Bool} ;
  Kind = CN ;
  Value = NP ;
  Name = NP ;
  Numeric = Card ;
lin
  OneSentenceDoc sent = mkText sent ;
  AddSentenceDoc doc sent = mkText doc (mkText sent) ;
  ConjSentence a b = mkS and_Conj a b ;
  FactSentence fact = mkS presentTense positivePol fact ;
  KindFact obj kind = mkCl obj.np (mkNP a_Det kind) ; --- sind ein Land
  PropertyFact obj prop = mkCl obj.np prop ;
  AttributeFact attr obj val = case obj.isPron of {
    True => mkCl (mkNP (mkDet obj.pron) attr) val ;
    _ => mkCl (mkNP the_Det (mkCN attr (mkAdv possess_Prep obj.np))) val
    } ;
  PropertyKind prop kind = mkCN prop kind ;
  ModifierKind kind mod = case mod.isAdv of {
    False => mkCN kind mod.rs ;
    True => mkCN kind mod.adv
    } ;
  NumericKindModifier num kind = mkModifier (mkAdv with_Prep (mkNP num kind)) ;
```

A **functor** opens **interfaces** instead of complete resources.

```
concrete FactsEng of Facts = FactsFunctor with
  (Syntax = SyntaxEng),
  (Symbolic = SymbolicEng),
  (Grammar = GrammarEng)
```

Functor instantiation.

```
concrete FactsGer of Facts = FactsFunctor with
  (Syntax = SyntaxGer),
  (Symbolic = SymbolicGer),
  (Grammar = GrammarGer)
```

```
concrete FactsFin of Facts = FactsFunctor with
  (Syntax = SyntaxFin),
  (Symbolic = SymbolicFin),
  (Grammar = GrammarFin)
```

restricted inheritance with an **exclude list**

```
concrete FactsFin of Facts = FactsFunctor - [AttributeFact]
with
  (Syntax = SyntaxFin),
  (Symbolic = SymbolicFin),
  (Grammar = GrammarFin)

** open (E=ExtendFin) in {

lin
  AttributeFact attr obj val = mkCl (mkNP (E.GenNP obj.np) attr) val ;
}
```

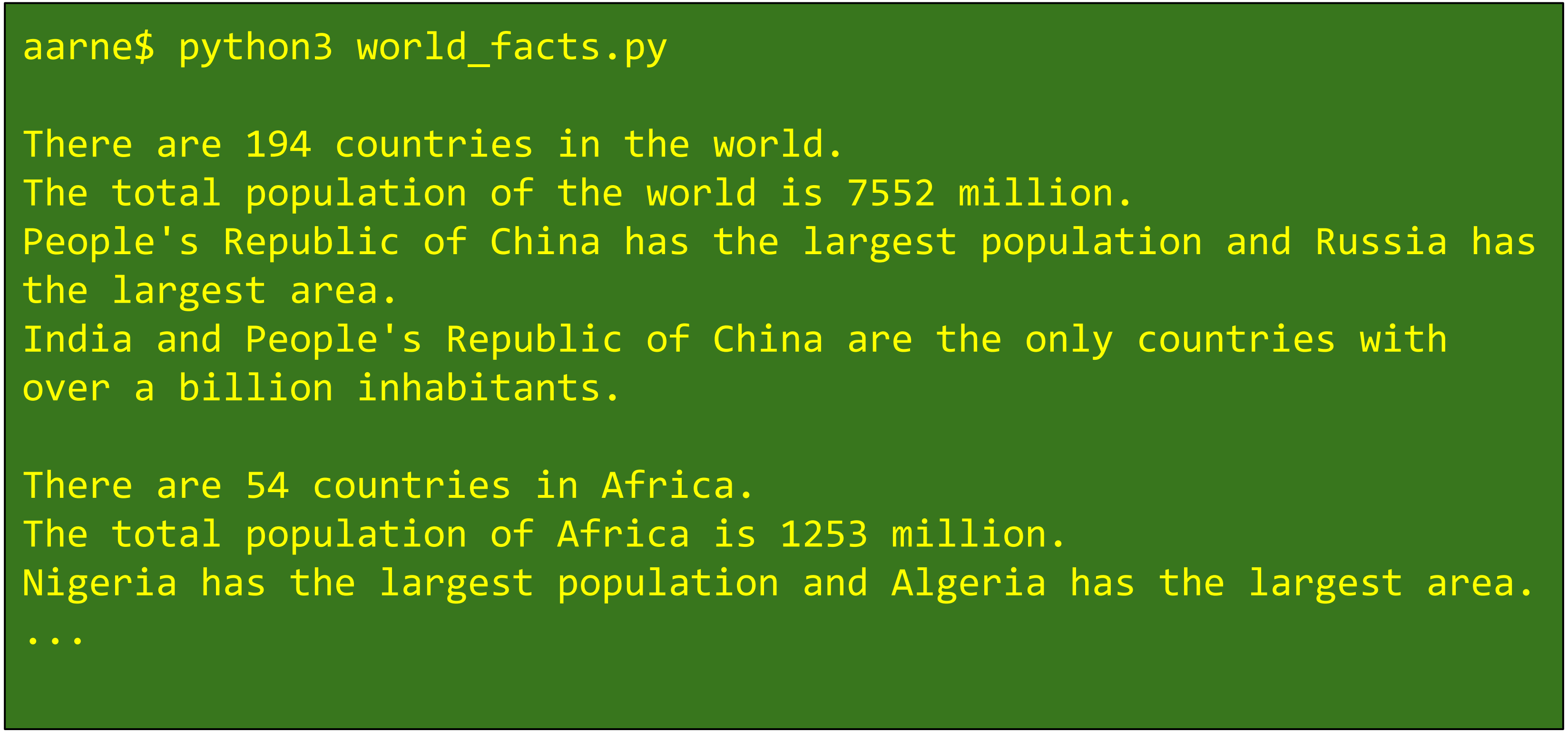
How GF helps, #4

Using a functor over the RGL API,

we can add rendering rules for a new language with just a few lines of code.

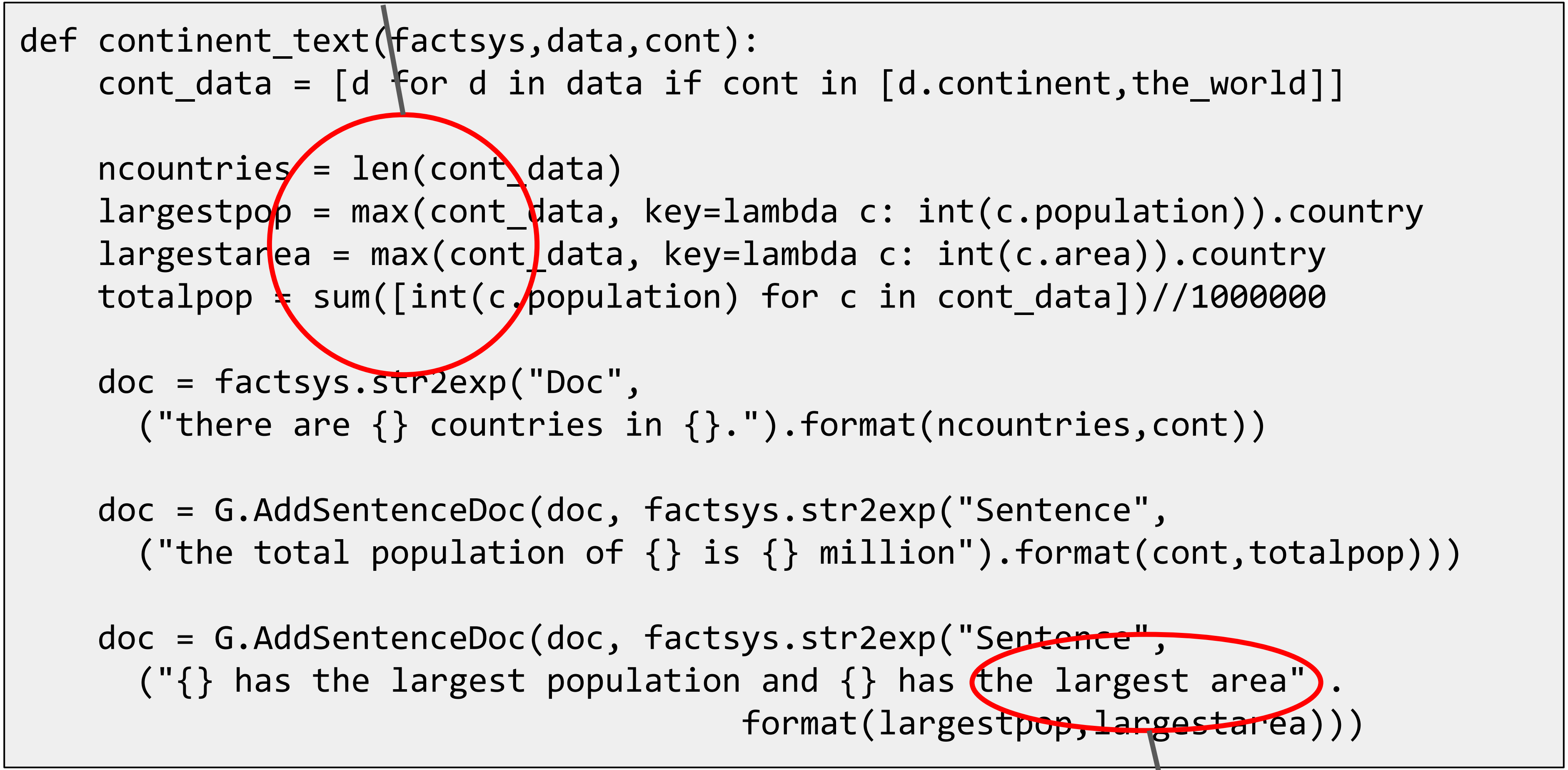
Stage 4:

Selecting content



```
aarne$ python3 world_facts.py

There are 194 countries in the world.
The total population of the world is 7552 million.
People's Republic of China has the largest population and Russia has
the largest area.
India and People's Republic of China are the only countries with
over a billion inhabitants.

There are 54 countries in Africa.
The total population of Africa is 1253 million.
Nigeria has the largest population and Algeria has the largest area.
...
```

https://github.com/aarneranta/NLG-examples/blob/main/doc/facts3/world_facts.py data aggregation



```
def continent_text(factsys,data,cont):
    cont_data = [d for d in data if cont in [d.continent,the_world]]

    ncountries = len(cont_data)
    largestpop = max(cont_data, key=lambda c: int(c.population)).country
    largestarea = max(cont_data, key=lambda c: int(c.area)).country
    totalpop = sum([int(c.population) for c in cont_data])//1000000

    doc = factsys.str2exp("Doc",
      ("there are {} countries in {}.").format(ncountries,cont))

    doc = G.AddSentenceDoc(doc, factsys.str2exp("Sentence",
      ("the total population of {} is {} million").format(cont,totalpop)))

    doc = G.AddSentenceDoc(doc, factsys.str2exp("Sentence",
      ("{} has the largest population and {} has the largest area").
                                     format(largestpop,largestarea)))
```

ellipsis

'X is the only K that P' is **correct by construction**

```
billions = [c.country for c in cont_data if int(c.population) > 1000000000]


if billions:
    doc = G.AddSentenceDoc(doc, factsys.str2exp('Sentence',
                object + ' with over a billion inhabitants'))
```

Who selects the content?

Stages 1 to 3:

- the SPARQL query selects the facts
- all facts are verbalized

Stage 4:

- function **world_facts.continent_text()** selects the data for each continent and for the world

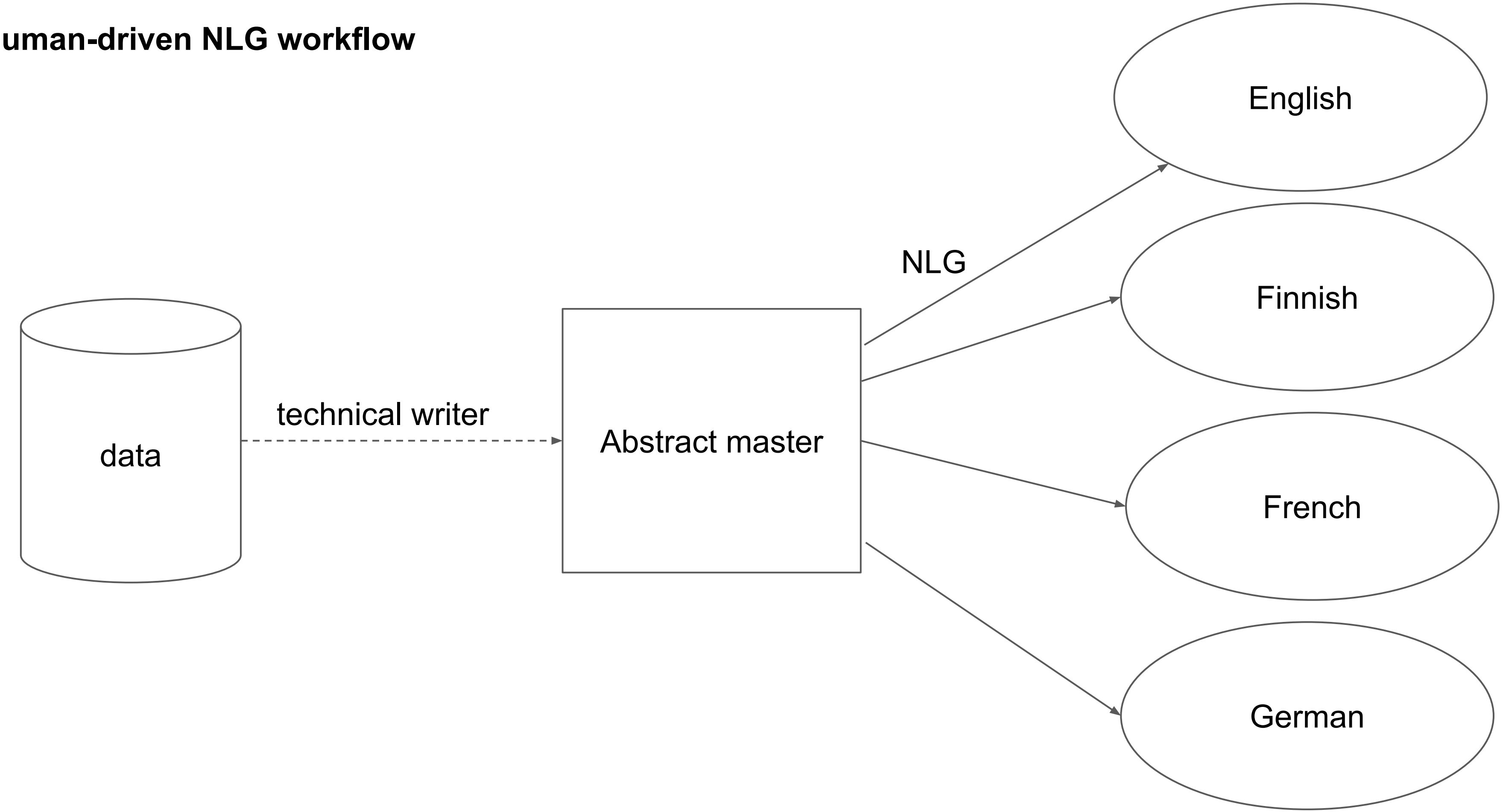
Next level: interactive selection

No algorithm can always select the most interesting data.

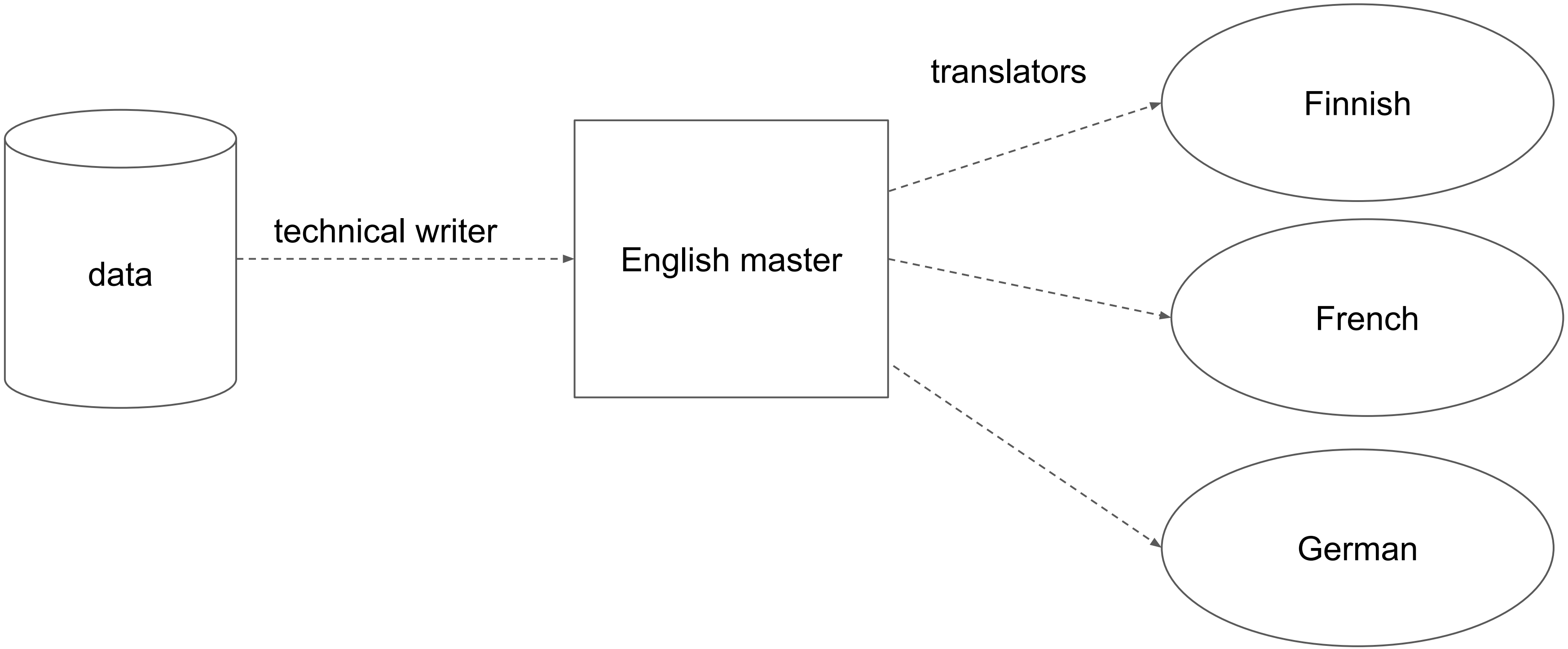
In the Abstract Wikipedia, the **authors** of documents must be able to select it.

Technically, this means that they **produce the master documents in abstract syntax**.

human-driven NLG workflow



compare with the traditional workflow



How to produce abstract master documents

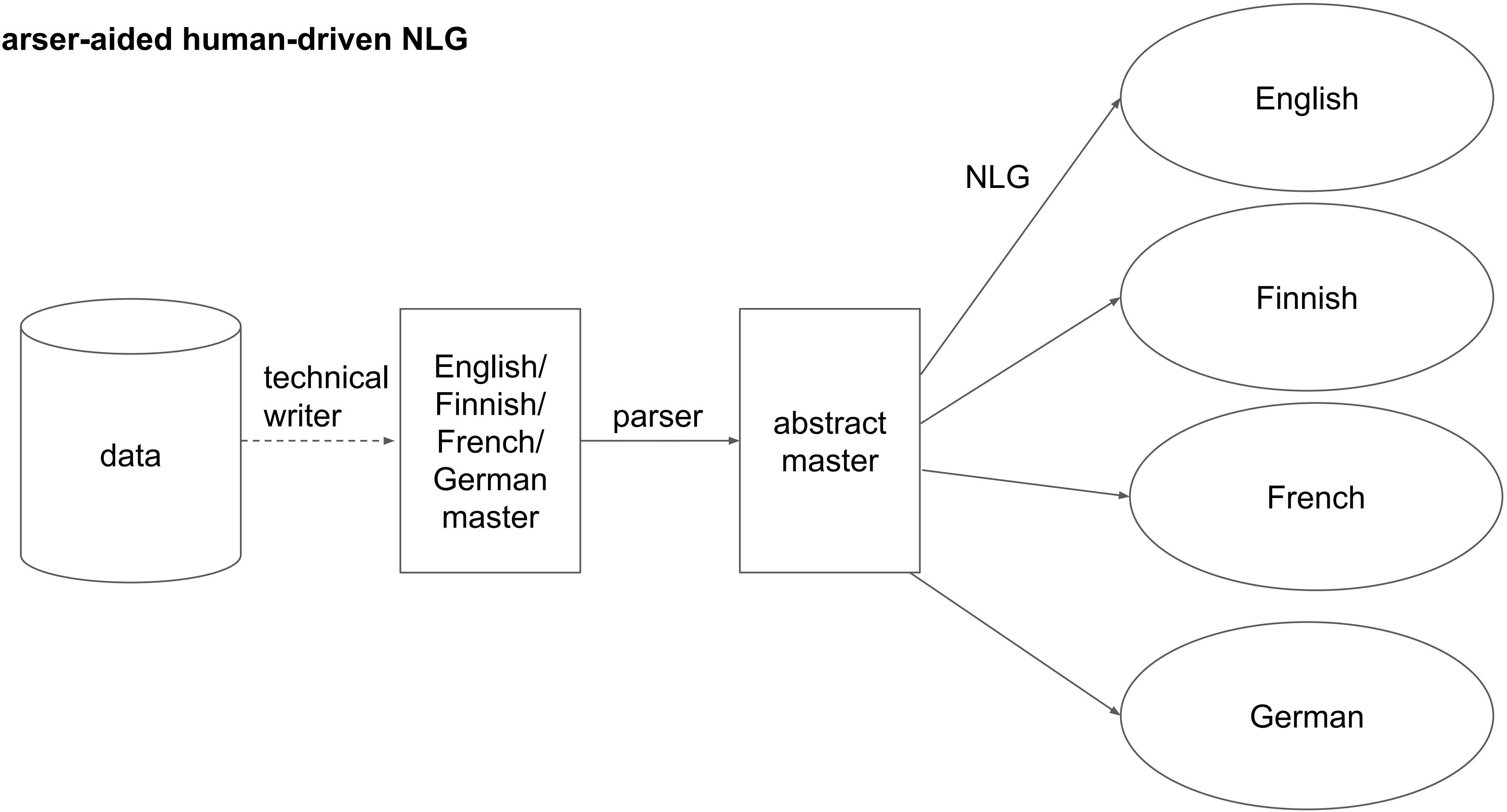
Write abstract syntax trees

Write programs that produce abstract syntax trees, e.g. **Wikifunctions**

Write natural language and let GF parse it

- the language must be covered by a GF grammar
- it can be any of the languages covered
- it supports **formal semantics** that can be **verified** w.r.t. Wikidata

parser-aided human-driven NLG



How GF helps, #5

Using the parser of GF,

we can extract abstract syntax from plain text such as legacy articles and author input.

Summary:

The NLG stages of Reiter & Dale

Reiter & Dale
content determination
discourse planning
sentence aggregation
lexicalization
referring expression generation
linguistic realization

Reiter & Dale	Stage 1
content determination	all facts in data
discourse planning	fact by fact
sentence aggregation	
lexicalization	data strings
referring expression generation	constants
linguistic realization	template

Reiter & Dale	Stage 1	Stage 2
content determination	all facts in data	
discourse planning	fact by fact	
sentence aggregation		
lexicalization	data strings	data labels
referring expression generation	constants	
linguistic realization	template	RGL

Reiter & Dale	Stage 1	Stage 2	Stage 3
content determination	all facts in data		
discourse planning	fact by fact		syntactic aggregation
sentence aggregation			syntactic aggregation
lexicalization	data strings	data labels	
referring expression generation	constants		pronouns
linguistic realization	template	RGL	RGL functor

Reiter & Dale	Stage 1	Stage 2	Stage 3	Stage 4
content determination	all facts in data			semantic aggregation
discourse planning	fact by fact		syntactic aggregation	
sentence aggregation			syntactic aggregation	
lexicalization	data strings	data labels		
referring expression generation	constants		pronouns	ellipsis
linguistic realization	template	RGL	RGL functor	

Research questions

NLG techniques extended

Reiter & Dale	Stage 1	Stage 2	Stage 3	Stage 4	to do
content determination	all facts in data			semantic aggregation	interactive authoring
discourse planning	fact by fact		syntactic aggregation		collect text patterns
sentence aggregation			syntactic aggregation		aggregation in all categories
lexicalization	data strings	data labels			WordNet, concept alignment
referring expression generation	constants		pronouns	ellipsis	definite descriptions
linguistic realization	template	RGL	RGL functor		language model optimization

GF scale-up required

From

- 40 languages, 100,000 abstract syntax functions

to

- 300 languages, 1,000,000 abstract syntax functions

Hence 2 orders of magnitude

GF scale-up required

From

- 40 languages, 100,000 abstract syntax functions

to

- 300 languages, 1,000,000 abstract syntax functions

Hence 2 orders of magnitude

To solve the **computational challenge**, we are looking at new models and algorithms, where

- the grammar is not in memory all the time but works like a database
- this is a major departure from the traditional theory of formal languages

GF scale-up required

From

- 40 languages, 100,000 abstract syntax functions

to

- 300 languages, 1,000,000 abstract syntax functions

Hence 2 orders of magnitude

To solve the **computational challenge**, we are looking at new models and algorithms, where

- the grammar is not in memory all the time but works like a database
- this is a major departure from the traditional theory of formal languages

To solve the **linguistic challenge**, we are looking at methods where

- more of the grammar can be extracted from data (existing texts, non-programmer informant input)
- support is given to resource grammars for low-resource languages with too little data

No resource grammar yet?

needed at Stage 2

```
AdverbEng.PrepNP
ConstructorsEng.ComplV2
ConstructorsEng.DetArtCard
ConstructorsEng.the_Det
NounEng.AdvCN
NounEng.DetCN
NounEng.IndefArt
NounEng.UseN
NounEng.UsePN
ParadigmsEng.regN
ParadigmsEng.regPN
SentenceEng.PredVP
StructuralEng.have_V2
StructuralEng.in_Prep
StructuralEng.possess_Prep
SymbolEng.IntPN
SymbolEng.SymbNum
SymbolicEng.mkSymb
VerbEng.CompAdv
VerbEng.CompNP
VerbEng.UseComp
```

additionally needed at Stages 3 and 4

```
AdjectiveEng.AdjOrd
AdjectiveEng.PositA
ConjunctionEng.BaseNP
ConjunctionEng.BaseS
ConjunctionEng.ConjNP
ConjunctionEng.ConjS
ConjunctionEng.ConsNP
ConjunctionEng.ListNP,
ExtendEng.CardCNCard
IdiomEng.ExistNP
IdiomEng.ExistNPAdv
NounEng.AdNum
NounEng.AdjCN
NounEng.DetNP
NounEng.DetQuant
NounEng.NumSg
NounEng.OrdSuperl
NounEng.PossPron
NounEng.RelCN
NounEng.UsePron
ParadigmsEng.mkAdN
```

```
ParadigmsEng.mkAdv
ParadigmsEng.mkOrd
ParadigmsEng.regA
PhraseEng.NoPConj
PhraseEng.NoVoc
PhraseEng.PhrUtt
PhraseEng.UttS
RelativeEng.RelVP
StructuralEng.and_Conj
StructuralEng.it_Pron
StructuralEng.somewhere_Adv;
StructuralEng.they_Pron
StructuralEng.with_Prep
VerbEng.CompAP
```

about 10% of the full RGL, normally doable in a couple of weeks by native speaker programmer

Conclusion

Abstract Wikipedia is a "Man on the Moon" project for NLG

- nothing of this size has been done before
- but we are confident that it is doable

The results of this work will be immediately available to billions of readers around the world, and we are tackling the issue of multilingual natural language generation on an unprecedented scale.

(Denny Vrandečić)

Thanks!
Any questions?