Developing an API for PanLex

Jonathan Pool

Computational Linguistics Group
University of Washington
30 January 2009
Topics

0. Introduction
1. Possibilities
2. Sample app
3. Under the hood
4. API benefits and costs
5. Best practices
6. Strategic choices
0. Introduction

PanLex: a panlingual lexical database
0. Introduction
0. Introduction

Database facts

Started 2006.
Versions:
  PostgreSQL/Linux with no inference (“PanLex”)
  SQL Server/Windows + peer-to-peer inference (“TransGraph”)
  SQL Server/Windows + centralized inference (“PanDictionary”)
Statistics by January 2009:
  12 million lexemes
  1,265 language varieties
  600 lexical resources
1. Possibilities

Games:

Panlingual “Free Rice”
1. Possibilities

Social networking:

Panlingual personals
1. Possibilities

Messaging:
1. Possibilities

Science:
1. Possibilities

Translation:

Translate Turkish words into 1260 languages!

Enter a word or phrase:
2. Sample app

Demo this translator

http://panlex.org/demo/trtur.html
3. Under the hood

Case 1

Client

What is the ID of "فراشة" in Turkish?

Server

It doesn’t exist.
3. Under the hood

Case 2

Client

- What is the ID of “çünkü” in Turkish?
- What language varieties can you translate 1355719 into?
- What translations from 1355719 into Ukrainian exist?

Server

- It is 1355719.
- Afrikaans; ...; עברית; ...; 日本語; ...; isiZulu
- адже; бо; внаслідок; оскільки; так як; тому; ...
3. Under the hood

Protocol: HTTP.
Server port: TCP 80.
Method: POST.
Content-type: XML.
Encoding: UTF-8.

Client → Server

POST http://panlex.org/p
Content-Type: application/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8"?>
<data><vr>00</vr><op>ex-0</op><lv>738</lv><et>çünkü</et></data>
3. Under the hood

Protocol: HTTP.
Server port: TCP 80.
Method: POST.
Content-type: XML.
Encoding: UTF-8.

Server → Client

HTTP/1.1 200 OK
Content-Type: application/xml; charset=utf-8

<?xml version="1.0" encoding="utf-8"?>
<data><vr>00</vr><op>ex-0</op><lv>738</lv><et>çünkü</et><ex>1355719</ex></data>
3. Under the hood

Request syntax

What is the ID of “çünkü” in Turkish?

<data>
  <vr>00</vr>
  <op>ex-0</op>
  <lv>738</lv>
  <et>çünkü</et>
</data>
3. Under the hood

Response syntax

The ID of “çünkü” in Turkish is 1355719.

<data>
  <vr>00</vr>
  <op>ex-0</op>
  <lv>738</lv>
  <et>çünkü</et>
  <ex>1355719</ex>
</data>
3. Under the hood

Error messages

<data>
  <err>In 2: Request content not UTF-8: 0fffd:000ef</err>
</data>

<data>
  <err>ex-0-DoArgs 3: No arg lv</err>
</data>
4. API benefits and costs

Benefits

Total utilization of the data.
Diversity of applications using the data.
Freedom from the obligation to do everything.
Recruitment of partners.
Revenue based on the total value of the resource (e.g., grants).
Visibility of the project.

Costs

Control of look and feel.
Control of uses of the data.
Risk of claims of infringement on intellectual-property rights.
Revenue from human visitors to one’s own service (e.g., advertising).
Freedom from the obligation to continue uninterrupted service provision.
5. Best practices

API pundits* say:

- Design APIs for simple, sequential request/response turns.
- Make rules easy and provide templates: Most developers are amateur or lazy.
- Offer multiple styles/formats/languages: Let each developer use familiar tools.
- Start basic: Let developers tell you what else they want.
- Don’t change a version: Never make developers change working apps.
- Write multiple demos: Force versatility on your API.
- Manage use: Human limits on traffic and abuse don’t apply.
- Follow general programming best practices: Document, modularize, hide, etc.

*Examples of API punditry:
Joshua Bloch (http://lcsd05.cs.tamu.edu/slides/keynote.pdf)
Nemetral (http://nemetral.net/2008/06/10/the-pursuit-of-apiness-part-1/)
Mashery (http://blog.mashery.com)
Alex Barnett (http://alexbarnett.net/blog/archive/2006/10/28/Web-API-Design-_2D00_-Keep-Some-of-it-Simple_2C00_-Stupid.aspx)
6. Strategic choices

- Protocol/style: DICT vs. REST vs. SOAP vs. XML-RPC vs. SQL.
- Server port: TCP 80 (HTTP) vs. TCP 2628 (DICT).
- Message medium: URI (GET) vs. standard input (POST).
- Encoding: UTF-8 re-encoding: no (multipart) vs. XML vs. URL.
- Content language: XML vs. JSON vs. token-sequence (DICT).
- Statefulness: yes (DICT) vs. no vs. partly (expiring details).
- Customization: stored settings.
- Utilization management: client keys, throttling, premium services.
- Limitations on resource-intensivity.
- Client hosting.
- Service elaboration.
- Aggregation of common service sequences.
- SQL as a service.
- Performance enhancement: caching, derivative tables, server clustering, EC2.
- Organization for durability.
6. Strategic choices

Popular protocols/styles

Dictionaries: DICT (RFC 2229*). Example:

C: DEFINE * shortcake

S: 150 2 definitions found: list follows
S: 151 "shortcake" wn "WordNet 1.5" : text follows
S: shortcake
S:   1. n: very short biscuit spread with sweetened fruit and usu.
S:      whipped cream
S: .
S: 151 "Shortcake" web1913 "Webster's Dictionary (1913)" : text follows
S: Shortcake
S:  \Short"cake`\, n.
S:  An unsweetened breakfast cake shortened with butter or lard,
S:  rolled thin, and baked.
S: .
S: 250 Command complete

*http://www.faqs.org/rfcs/rfc2229.html
6. Strategic choices

Popular protocols/styles

Everything else:

ProgrammableWeb* classifies 1,107 APIs by:
  Protocol or style: 64% REST, 22% SOAP, 7% JavaScript.
  Data format: 68% XML, 19% JSON, 7% RSS.

Of these, 57 are “reference” APIs. They are:
  Protocol or style: 54% REST, 46% SOAP.
  Data format: 81% XML, 18% JSON.

*http://www.programmableweb.com/apis/directory
6. Strategic choices

Popular message media

Standard input (less popular):

```
<data>
  <vr>00</vr>
  <op>ex-0</op>
  <lv>764</lv>
  <et>buôm buôm</et>
</data>
```

URI (more popular):

```
vr=00&op=ex-0&lv=764&et=buôm buôm
vr=00&op=ex-0&lv=764&et=b%c6%b0%c6%a1m b%c6%b0%e1%bb%9bm
```