PanLex: Using PostgreSQL to implement a massive word-translation graph

David Kamholz
PanLex project, The Long Now Foundation
kamholz@panlex.org
PanLex overview

• Non-profit project of The Long Now Foundation
• Mission: overcome language barriers to human rights, information, and opportunities
  ◦ focus on small, under-served language communities
• Main product: database connecting every word in every language
  ◦ 2,500 dictionaries
  ◦ 5,700 languages (of about 7,000 in the world)
  ◦ 25 million expressions (~words)
  ◦ 1.3 billion direct pairwise translations
  ◦ freely available
Presentation topics

• Database goals
• Database schema
• Word-translation query examples
  ◦ direct (attested) translations
  ◦ indirect (inferred) translations
  ◦ translation quality algorithms
• Strategies we’ve used to optimize performance
Database design goals

• Given a word in *any* language, get back translations in *any* other language

• Don’t make data model so rich or restrictive that it can’t support the limited data available for many smaller languages
  ◦ require only words in usual written form
  ◦ optional: part of speech, semantics, irregular forms, pronunciation, etc.
  ◦ make it possible to ingest widely available data (e.g. Wiktionary)

• Make it possible to infer new translations not directly attested in any source
## Database coverage: top 20 language varieties

<table>
<thead>
<tr>
<th>Language</th>
<th>num. expressions</th>
<th>Language</th>
<th>num. expressions</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>2,966,334</td>
<td>Uyghur (Arabic script)</td>
<td>396,771</td>
</tr>
<tr>
<td>Mandarin (Simplified)</td>
<td>1,661,698</td>
<td>Uyghur (Latin script)</td>
<td>350,252</td>
</tr>
<tr>
<td>Russian</td>
<td>1,203,275</td>
<td>Czech</td>
<td>338,493</td>
</tr>
<tr>
<td>French</td>
<td>691,415</td>
<td>Finnish</td>
<td>332,508</td>
</tr>
<tr>
<td>German</td>
<td>686,887</td>
<td>Portuguese</td>
<td>298,375</td>
</tr>
<tr>
<td>Mandarin (Pinyin)</td>
<td>637,854</td>
<td>Polish</td>
<td>287,792</td>
</tr>
<tr>
<td>Japanese</td>
<td>592,244</td>
<td>Dutch</td>
<td>284,054</td>
</tr>
<tr>
<td>Spanish</td>
<td>561,276</td>
<td>Arabic</td>
<td>278,293</td>
</tr>
<tr>
<td>Italian</td>
<td>488,153</td>
<td>Esperanto</td>
<td>265,415</td>
</tr>
<tr>
<td>Mandarin (Traditional)</td>
<td>469,780</td>
<td>Hindi</td>
<td>264,541</td>
</tr>
</tbody>
</table>
PanLex database concepts 1

- **language**: the set of linguistic varieties designated by a single ISO 639 three-letter code (e.g., `eng` = English, `cmn` = Mandarin, etc.)
- **language variety**: particular variety of a language (distinguished by dialect, script, etc.)
- **expression**: string of characters in a language variety, representing a word or word-like phrase (“try”, “try out”, “trial and error”)
- **resource**: anything that documents equivalences among expressions (dictionary, thesaurus, thematic word list, database, etc.)
- **source**: logical chunk of a resource, as represented in PanLex (if not whole resource, could be each direction of a bilingual dictionary)
PanLex database concepts 2

- **meaning**: single set of inter-translated expressions from a source (always belongs to **one** source!)
- **denotation**: pairing of an expression with a meaning in a source
- **translation**: two expressions that are (arguably) equivalent
  - direct: explicitly attested in some PanLex source
  - indirect: inferred by combining multiple PanLex sources
- **translation examples**
  - English “couch” = English “sofa”
  - English “cat” = Spanish “gato” = German “Katze” (3 pairwise translations)
Database schema illustration
Source table definition

CREATE TABLE source (  
id serial PRIMARY KEY,  
label text NOT NULL UNIQUE,  
    -- standardized human-readable label e.g. ‘eng-spa-Smith’  
quality smallint NOT NULL,  
    -- ranges from 0 to 9  
grp integer NOT NULL REFERENCES source(id)  
    -- groups sources together from the same resource  
);
Expression table definition

CREATE TABLE expr (  
id serial PRIMARY KEY,  
langvar integer NOT NULL REFERENCES langvar(id),  
txt text NOT NULL,  
UNIQUE (txt, langvar)  
);
Meaning and denotation table definitions

CREATE TABLE meaning ( -- translation set from a source
    id serial PRIMARY KEY,
    source integer NOT NULL REFERENCES source(id)
);

CREATE TABLE denotation ( -- links expression and meaning
    id serial PRIMARY KEY,
    meaning integer NOT NULL REFERENCES meaning(id),
    expr integer NOT NULL REFERENCES expr(id),
    UNIQUE (meaning, expr)
);
Denotationx table definition (denormalized)

CREATE TABLE denotationx (  -- denormalized denotation
    id integer NOT NULL REFERENCES denotation(id),
    meaning integer NOT NULL,  -- denotation.meaning
    expr integer NOT NULL,  -- denotation.expr
    langvar smallint NOT NULL,  -- denotation.expr.langvar
    source smallint NOT NULL,  -- denotation.meaning.source
    grp smallint NOT NULL,  -- denotation.meaning.source.grp
    quality smallint NOT NULL,  -- denotation.meaning.source.quality
    UNIQUE (meaning, expr)
);
Schema design decisions

• Meaning-denotation-expression design creates some level of indirection, but allows translation sets to be arbitrarily large
  ◦ translation set with 500 expressions has 124,750 undirected pairs
  ◦ gets costly to derive and store each pair

• PanLex concept of meaning is aspirational, not literal (confusingly)
  ◦ two sources documenting an equivalence between English “cat” and Spanish “gato” will create *two different* PanLex meanings, despite having the same linguistic meaning
  ◦ because can’t always be sure which PanLex meanings have the same linguistic meaning: which did dictionary author intend? think of words like “bank” (financial institution vs. edge of river), “play” (verb vs. noun meaning theater piece)
  ◦ aspiration to eventually merge PanLex meanings that share the same linguistic meaning (not easy!)
How would you write a translation query?
Direct translation query
English “eggplant” → Spanish

SELECT DISTINCT expr.id, expr.txt
FROM expr
JOIN denotationx AS d ON d.expr = expr.id
JOIN denotationx AS d_src ON d_src.meaning = d.meaning AND d_src.expr != d.expr
WHERE expr.langvar = 666 AND d_src.expr IN (SELECT expr.id FROM expr WHERE expr.langvar = 187 AND expr.txt = 'eggplant');

• NB: langvar 666 = Spanish, 187 = English
Direct translation query

English “eggplant” → Spanish

<table>
<thead>
<tr>
<th>id</th>
<th>txt</th>
</tr>
</thead>
<tbody>
<tr>
<td>654994</td>
<td>berenjena</td>
</tr>
<tr>
<td>8368676</td>
<td>aubergine</td>
</tr>
<tr>
<td>8374926</td>
<td>manzana de amor</td>
</tr>
<tr>
<td>20909249</td>
<td>solanum melongena</td>
</tr>
<tr>
<td>23225336</td>
<td>color berenjena</td>
</tr>
</tbody>
</table>

- Results are good, but unordered and don’t know which is “better”
Direct translation query with quality
English “eggplant” → Spanish

```
SELECT expr.id, expr.txt, grp_quality_score(array_agg(dgrp),
    array_agg(d.quality)) AS trans_quality
FROM expr
JOIN denotationx AS d ON d.expr = expr.id
JOIN denotationx AS d_src ON d_src.meaning = d.meaning AND
    d_src.expr != d.expr
WHERE expr.langvar = 666 AND d_src.expr IN (SELECT expr.id FROM
    expr WHERE expr.langvar = 187 AND expr.txt = 'eggplant')
GROUP BY expr.id
ORDER BY trans_quality DESC;
```
Implementation of `grp_quality_score`

```
CREATE FUNCTION grp_quality_score(grp integer[], quality smallint[]) RETURNS integer LANGUAGE sql IMMUTABLE AS $$
SELECT sum(max_quality)::integer
FROM (SELECT max(quality) AS max_quality FROM (SELECT * FROM unnest(grp, quality) AS u(grp, quality)
GROUP BY grp)
) b
$$;
```
## Direct translation query with quality: result

English “eggplant” → Spanish

<table>
<thead>
<tr>
<th>id</th>
<th>txt</th>
<th>trans_quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>654994</td>
<td>berenjena</td>
<td>65</td>
</tr>
<tr>
<td>23225336</td>
<td>color berenjena</td>
<td>7</td>
</tr>
<tr>
<td>20909249</td>
<td>solanum melongena</td>
<td>3</td>
</tr>
<tr>
<td>8368676</td>
<td>aubergine</td>
<td>2</td>
</tr>
<tr>
<td>8374926</td>
<td>manzana de amor</td>
<td>2</td>
</tr>
</tbody>
</table>
What about a less typical pair of languages?

- Probably don’t have an Irish Gaelic–Moor dictionary...
Indirect translation query
Irish Gaelic “madra” (dog) → Moor

SELECT expr.id, expr.txt, grp_quality_expr_score_geo2(array_agg(d.grp),
    array_agg(d_src.grp), array_agg(d.quality), array_agg(d_src.quality),
    array_agg(d2.expr)) AS trans_quality
FROM expr
JOIN denotationx AS d ON d.expr = expr.id
JOIN denotationx AS d2 ON d2.meaning = d.meaning AND d2.expr != d.expr
JOIN denotationx AS d3 ON d3.expr = d2.expr
JOIN denotationx AS d_src ON d_src.meaning = d3.meaning AND d_src.grp
    != d.grp AND d_src.expr != d.expr AND d_src.expr != d3.expr
WHERE expr.langvar = 886 AND d_src.expr IN (SELECT expr.id FROM expr
    WHERE expr.langvar = 238 AND expr.txt = 'madra')
GROUP BY expr.id
ORDER BY trans_quality DESC;
Implementation of `grp_quality_expr_score_geo2`

CREATE FUNCTION grp_quality_expr_score_geo2(grp1 integer[], grp2 integer[], quality1 smallint[], quality2 smallint[], expr2 integer[]) 
RETURNS integer LANGUAGE sql IMMUTABLE AS $$
SELECT round(sum(sqrt(b.quality1*b.quality2)))::integer 
FROM ( 
    SELECT max(a.quality1) AS quality1, max(a.quality2) AS quality2 
    FROM ( 
        SELECT * FROM unnest(grp1, grp2, quality1, quality2, expr2) AS 
            u(grp1, grp2, quality1, quality2, expr2) 
        ) a 
    GROUP BY a.grp1, a.grp2, a.expr2 
    ) b $$;
Indirect translation query: result
Irish Gaelic “madra” (dog) → Moor

<table>
<thead>
<tr>
<th>id</th>
<th>txt</th>
<th>trans_quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>7409488</td>
<td>auna</td>
<td>2692</td>
</tr>
<tr>
<td>7409578</td>
<td>savaʔu</td>
<td>6</td>
</tr>
<tr>
<td>18744101</td>
<td>maʔa</td>
<td>5</td>
</tr>
</tbody>
</table>
**Indirect translation: top intermediate languages**  
Irish Gaelic “madra” (dog) $\rightarrow$ Moor

<table>
<thead>
<tr>
<th>langvar</th>
<th>times used</th>
<th>langvar</th>
<th>times used</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>44</td>
<td>hrvatski</td>
<td>15</td>
</tr>
<tr>
<td>Esperanto</td>
<td>34</td>
<td>Malti</td>
<td>14</td>
</tr>
<tr>
<td>Tagalog</td>
<td>19</td>
<td>hornjoserbšćina</td>
<td>12</td>
</tr>
<tr>
<td>bokmål</td>
<td>17</td>
<td>Iloko</td>
<td>12</td>
</tr>
<tr>
<td>català</td>
<td>17</td>
<td>føroyskt</td>
<td>12</td>
</tr>
<tr>
<td>eesti</td>
<td>17</td>
<td>bosanski</td>
<td>11</td>
</tr>
<tr>
<td>latviešu</td>
<td>17</td>
<td>Volapük</td>
<td>10</td>
</tr>
<tr>
<td>brezhoneg</td>
<td>16</td>
<td>basa Jawa</td>
<td>9</td>
</tr>
<tr>
<td>bahasa Indonesia</td>
<td>16</td>
<td>Bahasa Malaysia</td>
<td>9</td>
</tr>
<tr>
<td>slovenčina</td>
<td>15</td>
<td>isiZulu</td>
<td>9</td>
</tr>
</tbody>
</table>
Performance observations

• Denormalized denotation\textsubscript{x} table is a big win overall
  ◦ reduces number of joins (especially in indirect translation queries)
  ◦ uses less memory
  ◦ produces better query plans

• Translation queries benefit from having working set (\texttt{expr} and denotation\textsubscript{x} tables and indexes) loaded into memory
  ◦ single dedicated server with 128GB of RAM hosts PanLex database
  ◦ extra memory at lower price point (no cloud) is worth the loss of flexibility/scalability
  ◦ (keep in mind: PanLex is a small project with limited staff)
By the way, you can do more than just translation with PanLex…

- Generate fake words in a language using a Markov-chain model
- English examples:
  - hired mullet
  - adjustache
  - nuclear souffle
  - predestructural struction
  - garbling port
  - telephantability
  - pepperonism
  - vermicrosoft driverian