Domain-specific knowledge graphs construction: challenges and opportunities

Omar Alonso

14-Sept-2021

Disclaimer

The views, opinions, positions, or strategies expressed in this talk are mine and do not necessarily reflect the official policy or position of my employer.

Outline

Introduction

Some domain examples

Content

Infrastructure

Applications

Conclusion

Introduction

Introduction

Interest in KGs

Many examples

Google, Microsoft, Amazon, Yago, DBpedia, Wikidata, UnitProt

Lots of buzzwords

Definitions

Organizing data as nodes and edges
KG is a repository of entities, types, and relationships
KG is data
KG evolves and needs maintenance

Why KGs?

Semantic search

Going beyond 10-blue links

Understanding queries and documents

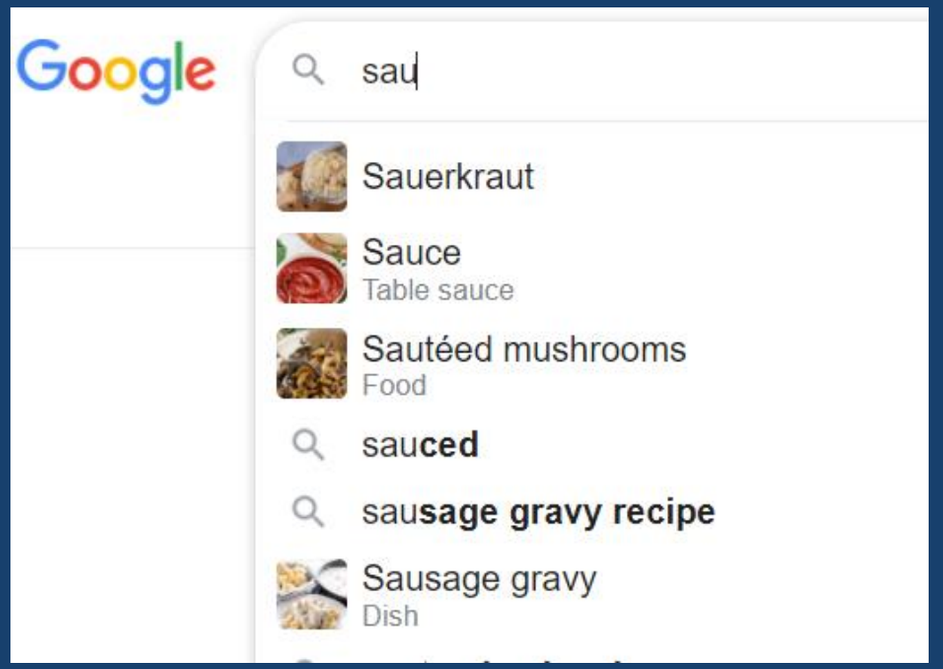
Question-answering

Entity retrieval

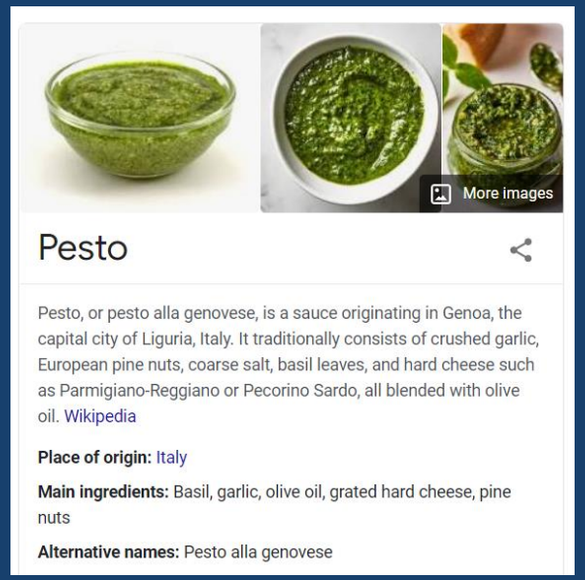
Ads

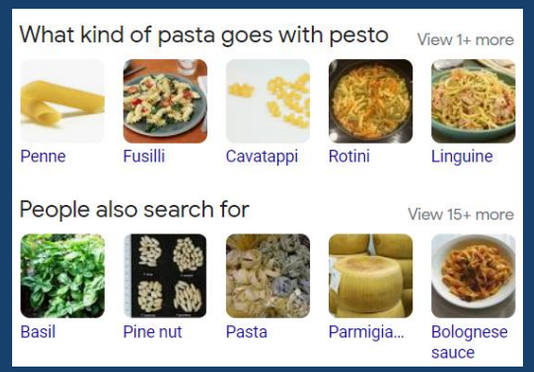
Data cleaning

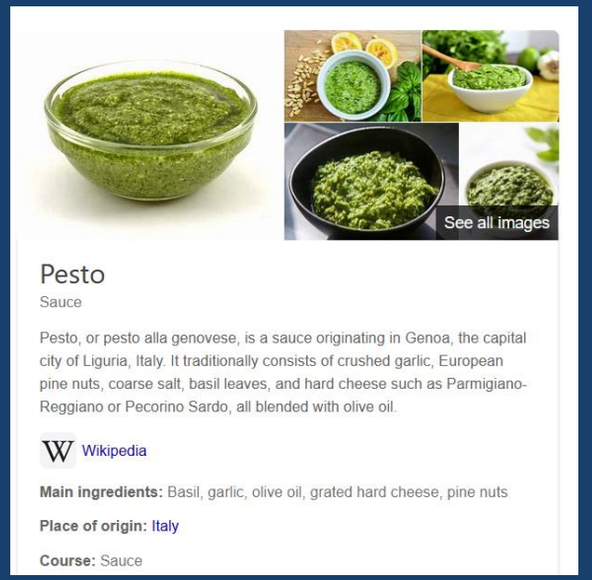
Example - Autocomplete



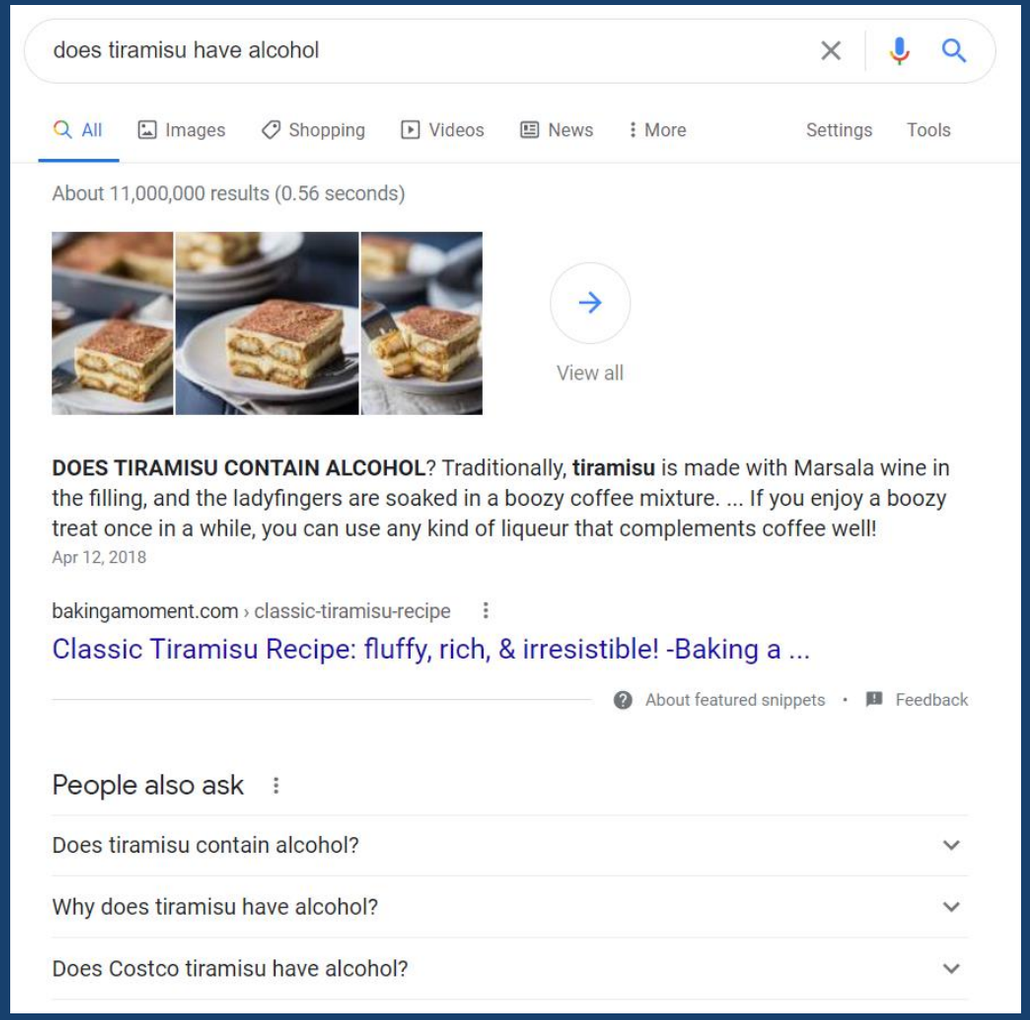
Example - Entity cards

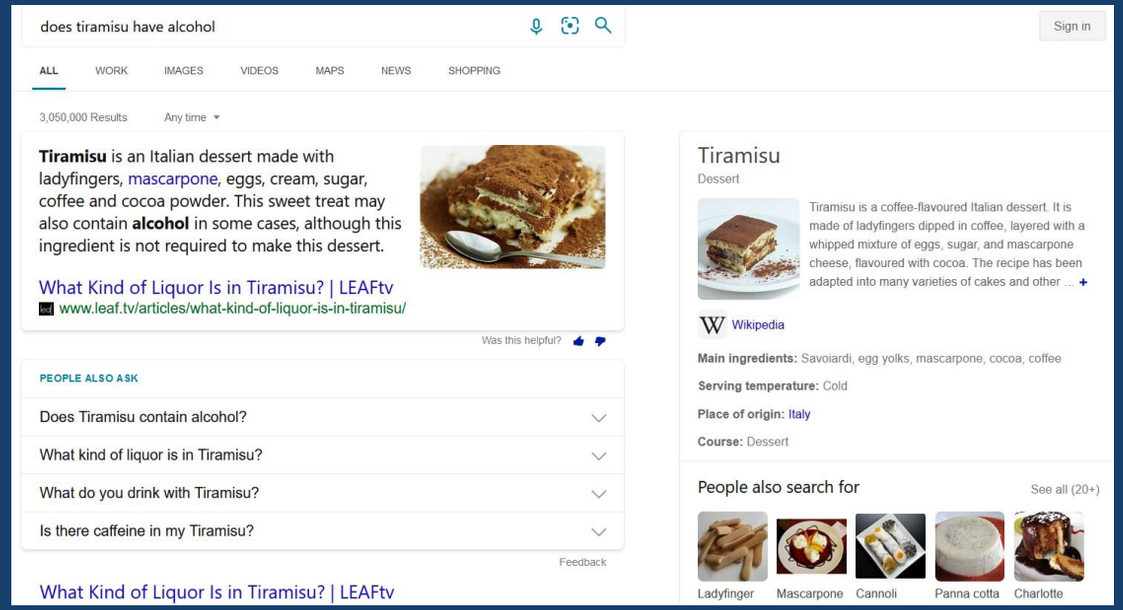






Example - Answers





Opportunities

How to start?

Most of research on KGs/KBs use Wikipedia

Benefits: easy to read, easy to parse, Wikipedians

Drawbacks: coverage, outdated content, bias

What to do when there is no Wikipedia?

Many design options

Construction and production

Focus of this talk

Content, infrastructure and applications

Iterative development

Domain specific KGs

Social Knowledge Graph

Input: Twitter firehose

Output: a knowledge graph

Components

Links

Topics

Entities (e.g., people, organizations, places)

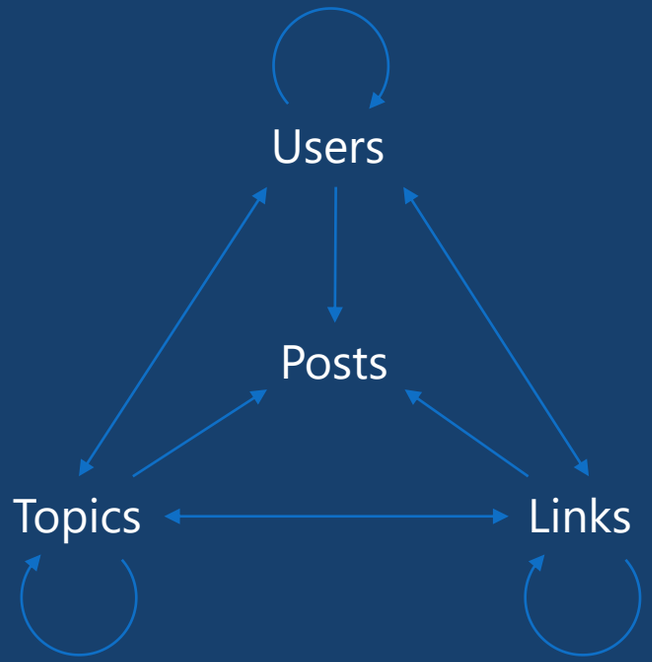
Time

Focus on high-quality content

Relevant content from trusted users on good topics

O. Alonso, V. Kandylas, S. Tremblay. "How it Happened: Discovering and Archiving the Evolution of a Story Using Social Signals", JCDL 2018
O. Alonso, V. Kandylas, S. Tremblay. “Social Knowledge Graph Explorer". SIGIR 2019.

SKG Core Schema



- Select best subsets
  - Users: verified + trusted
  - Links: top + viral + trending
  - Topics:
    - Hashtags: top + trending
    - Entities: top + trending
    - Cashtags: top
    - Ngrams: top + trending + link social signatures
  - Posts: top 5 posts per selected user/link/topic

Story evolution

Table of contents

Hit-list clustering

Story

Timeline generation

Related stories

Pivots

Related hashtags and topics

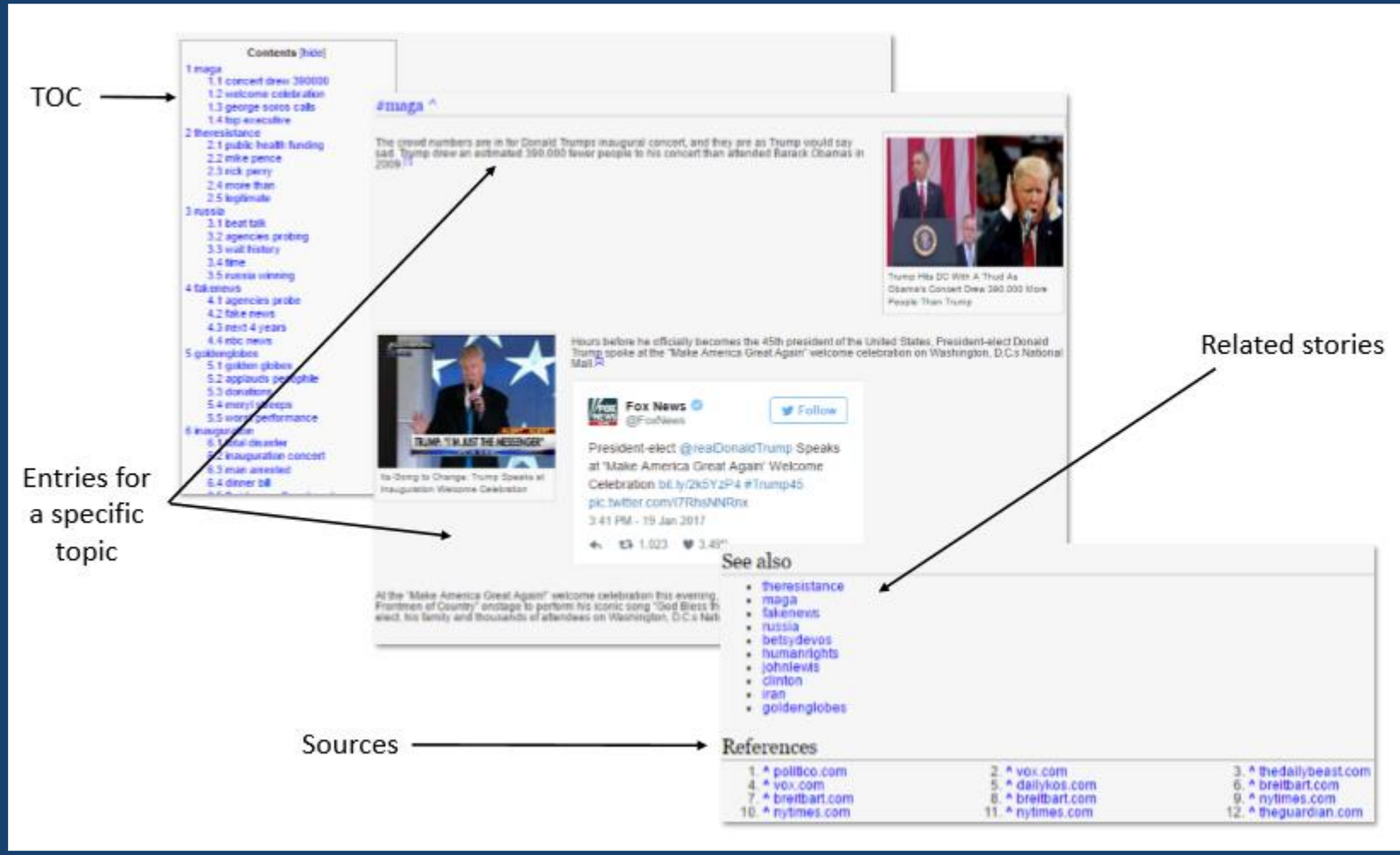
Sources

Domains

Queries

Derived from the story

Annotations



Brands, products, and categories

Scenarios

Query for brand (Microsoft) and return products (Office, Surface, Windows)
Query for product (jeans) and return brands (Calvin Klein, Levi's)
Query for category (smartphones) and return products (iPhone, Galaxy, Pixel)

Products can be items for sale or services

Adidas samba, insurance, food delivery

Competitors

Related products and brands

Product families

O. Alonso, V. Kandylas, R. Iyer. "Unsupervised construction of a product knowledge graph" eCom@SIGIR 2019

Challenges

Very dynamic domain (brands and products appear/disappear)

Lack of major sources with clean brand/product data

Hard to define and detect products

Distinction between brand and product is sometimes blurred

Retailers don’t always provide clean data

Approach

Unsupervised

Focused on data quality and simplicity

Generate brands using data fusion and voting

Tag brands with categories

Generate products using different techniques

Healthcare domain

Scientific medical knowledge

Existing taxonomies and data sources

Examples

SNOMED (Systematized Nomenclature of Medicine)
RxNorm (medications available on the US market)
MeSH (Medical Subject Headings)

Challenges

Very sensitive data

EMR (Electronic Medical Records)

Clinical relevance

Vocabulary mismatch

Patient describing a symptom

MDs describing a diagnosis

Data labeling and curation

How to bootstrap?

No single approach to build a KG

Research and engineering problems

Iterative development cycle

Content

Infrastructure

Applications

Content

Input sources

Data

Wikipedia, catalogs, web pages, query logs, databases etc.

Importance of top-tier sources

Authoritative content, high coverage, clean representation

Domain specific

Pre-existing categorization

Potentially useful

Alignment

Entity discovery

NER detects mentions of entities and assigns types

Dictionaries

ML

CRF, LSTM

Embeddings

Taxonomies from catalogs and user behavior

Not always available

Attributes and relationships

Pattern-based

Regex

Rule-base extraction

Extraction from semi-structured content

DOM trees

Web tables

Information extraction

Relationships and attributes

SPO

Subject-Predicate-Object

Example

<Tom Brady, place of birth, San Mateo>
<Tom Brady, member of sports team, Tampa Bay>
<Tom Brady, occupation, American football player>
<fettuccine, subclass of, pasta>
<fusilli, subclass of, pasta>
<linguine, subclass of, pasta>
<paella, country of origin, Spain>
<paella, has ingredient, chicken>
<paella, has ingredient, rice>

Infrastructure

Data models

Direct edge-labeled graphs

RDF is an example

Graph dataset

Set of named graphs. Each named graph is a pair (graph id, graph)

Property graphs

Allows a set of (property, value) pairs and a label to be associated with nodes/edges

Common in graph databases

Data access

Querying

SPARQL
SQL

Raw data

Key, values

Data workflow

Ability to generate a KG from scratch
Orchestration of sources and data generation

Materialization

Publish high quality data

Search & Browse UI

KG curation

Data quality

Human in the loop

KG life cycle

Provenance

Versioning

Maintenance

Feedback loops

Reliability scores

KGs are derived from multiple sources

Difficult to curate by hand

Test a slice of KG and produce a score

Numerical
Constraint based
Clustering
Anomaly detection
Provenance

White vs black box

Interpretability

Representation

Unique problem

Entities in KG have no textual representation, apart from their names
We can run SPARQL queries but how do we add the IR part?

Predicate folding

Build a textual representation for each entity by considering all triples
Grouping predicates together into a small set of predefined categories
From SPOs triples to a structured document

Predicate folding - example

```
<spaghetti carbonara, instance_of, recipe>
<spaghetti carbonara, has_ingredient, spaghetti>
<spaghetti carbonara, has_ingredient, pancetta>
<spaghetti carbonara, has_ingredient, eggs>
<spaghetti carbonara, has_ingredient, parmesan>
<spaghetti carbonara, recipe_cuisine, italian cuisine>
<spaghetti carbonara, serving_size, 4>
<spaghetti carbonara, calories, 510>
<spaghetti carbonara, cook_time, 25min>
```

Name	spaghetti carbonara
Ingredients	Spaghetti, pancetta, eggs, parmesan
Attributes	italian cuisine, serves 4, calories 510, cook time 25min
Related entities	spaghetti aglio e olio, fettuccine alfredo

Entity linking

Recognizing entity mentions in text and linking them to the corresponding entries in a KG

Assume a KG with existing entities

Mention detection

Identification of text snippets that can potentially be linked to entities

Candidate selection

Ranked list of candidate entities is generated for each mention

Disambiguation

The best entity (or none) is selected for each mention using context (if available)

Ranking problem

Applications

Some scenarios

Search

Augment search, query understanding, user intent

Ads

Keyword bidding on nodes and relationships
Example: competitors for pasta brand (Barilla, Colavita)

Recommendations

Recommend products and recipes to users

Question-answering

`<cheesecake, has_ingredient, egg>`

This triplet can be used to answer queries like “does cheesecake have eggs”

Document retrieval

Preprocessing

Documents are preprocessed with EL + additional information obtained from KG

Query annotation

Query processed with EL

Expansion

KG feedback: query is issued against an index of a KG in order to retrieve related entities

Corpus-based feedback

Entity retrieval

Field search retrieval

Linear combination of matching functions

Can use LTR to learn weights

$$\text{score} = w_1 * match(f_1, q) + w_2 * match(f_2, q) + \ldots + w_i * match(f_i, q)$$

Summary

Focus on the utility of the KG data first

Decide the minimum infra required

Approach

Identify a clear use case
Select a high-quality data set
Ingest data and generate RDF
Store in graph database
Materialize data for consumption
Serve a simple application

Iterate

Conclusion

Active area of work in industry and academia

Combination of many techniques

Importance of high-quality sources

Identify the minimum requirements for infra

Have a clear use case in mind

An imperfect KG is still useful

Thanks!