Urban Perception Extraction From Texts Shared on Social Media: Framework and Applications

Frances Albert Santos

Supervisor: Prof. Leandro Aparecido Villas

Co-supervisor: Prof. Thiago Henrique Silva (UTFPR)

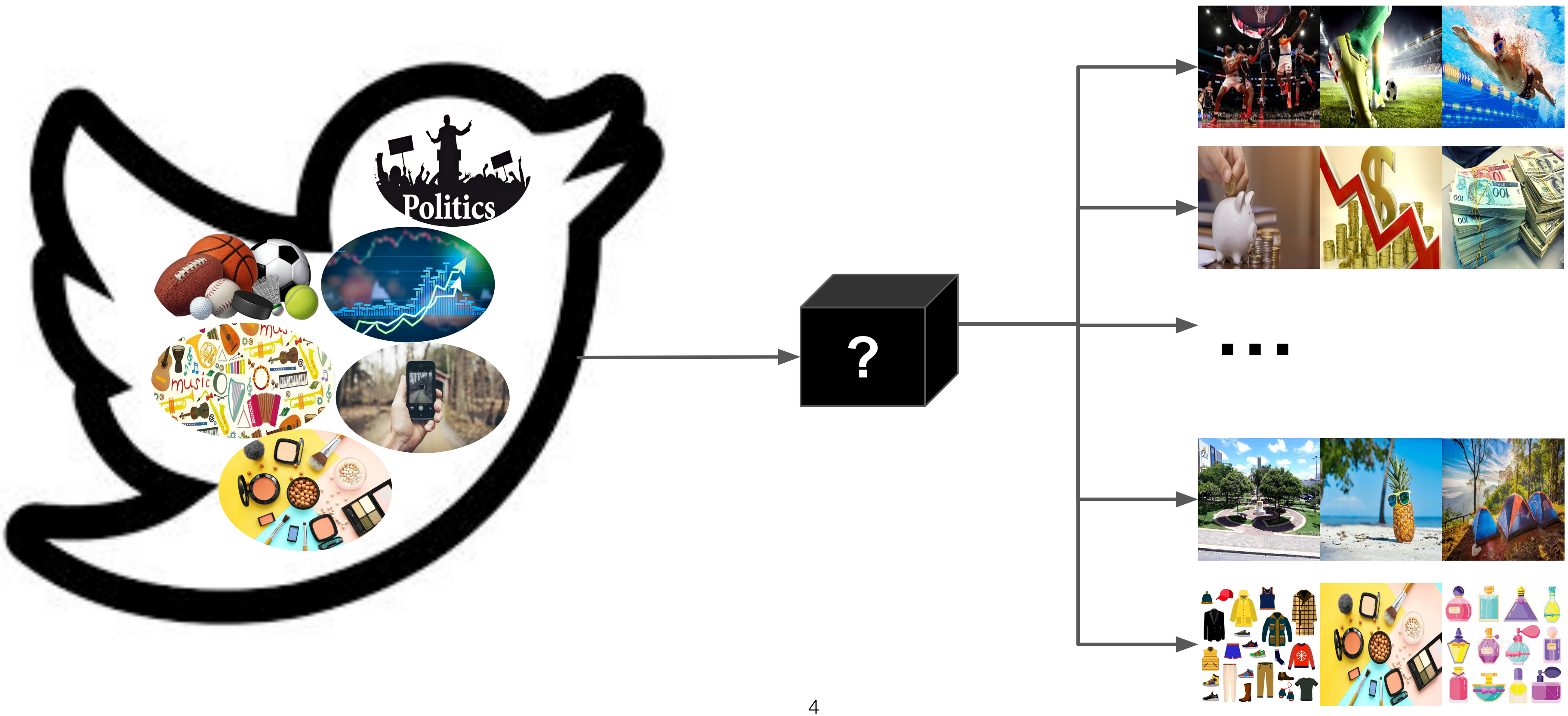
Campinas

October 27, 2021

Natural Language Understanding
NLU

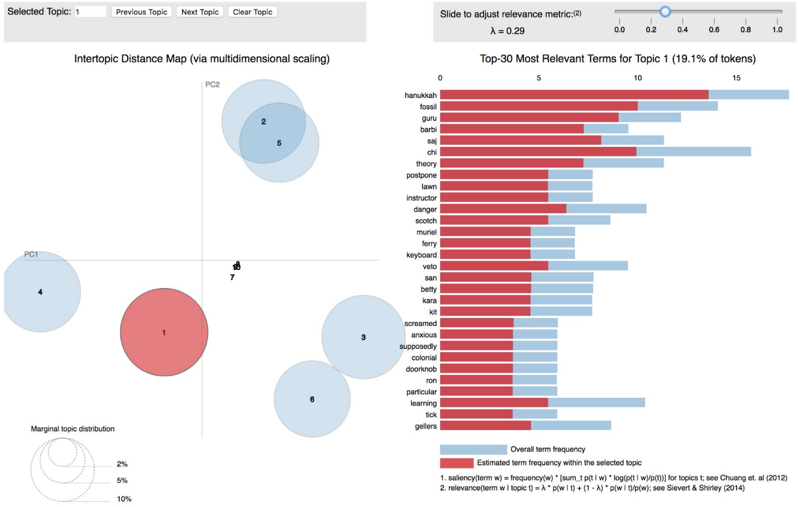
Extract useful urban perceptions using social media content to help better understanding urban areas and leverage new services and applications

Problem 1: How can we extract urban perceptions from social media data (natural language texts)?



Attempt 1: Keyword-Based Topic Modeling (2016)

- Define a subset of keywords related to urban perceptions.
- Using a Topic Modeling approach (e.g., LDA), we could group data into topics and identify the most relevant words for each topic

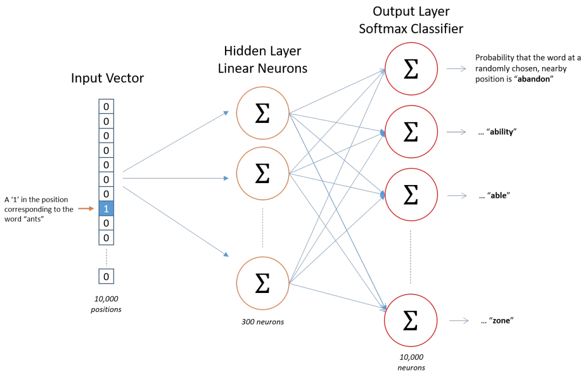


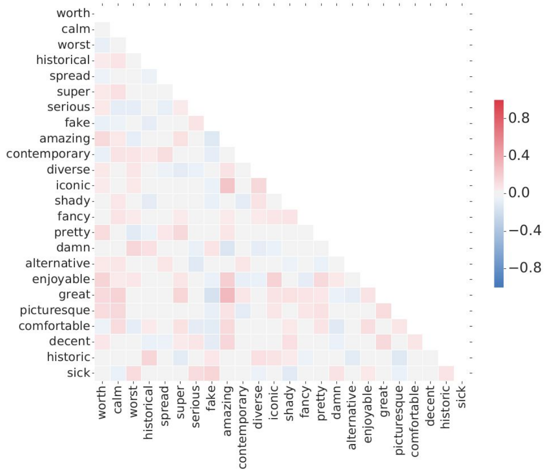
Problem: some words can be used in different contexts: shooting, crime, great, etc. Thus, a huge amount of extracted data was not related to urban perception.

shooting

Attempt 2: Unsupervised Learning

- Using a Word2Vec model, trained with users' reviews shared on crowdsourcing systems, to estimate the similarity of a tweet as the corpus





Problems:

- Small corpus (54,612)
- Noisy data
- Hard to distinguish among urban perceptions (traffic, safety, etc)

A considerable amount of extracted data still was not related to urban perception.

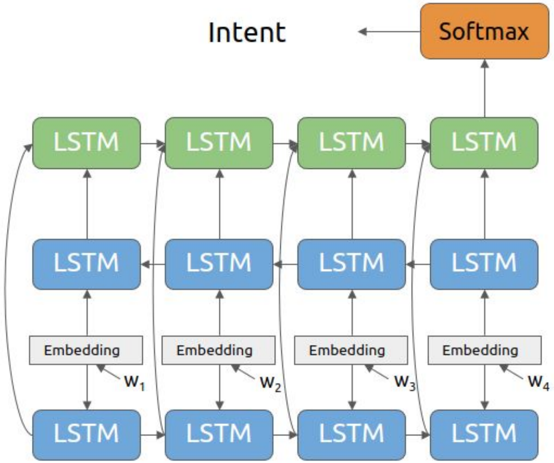
Attempt 3.1: Intent Recognition

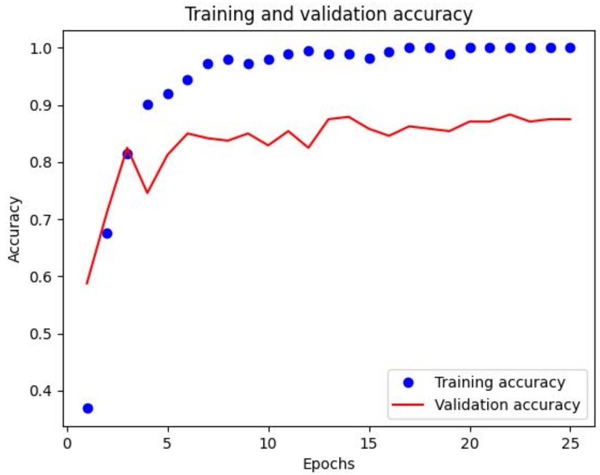
- Manually label a tweet data set to train a supervised model

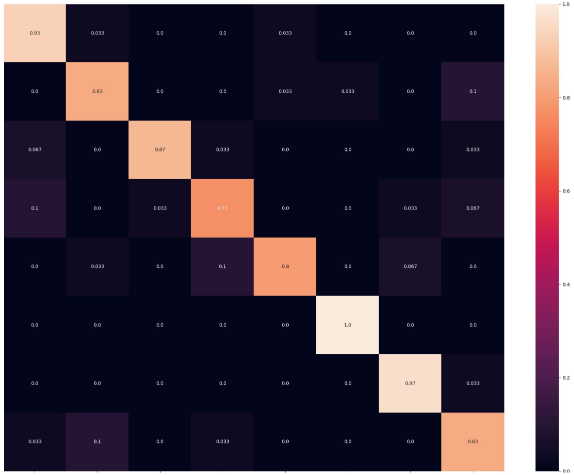
  8 categories (intents); 1,200 samples; 150 for each intent; 120/30 for training/testing

- Create a RNN model (BiLSTM + LSTM)

Problems:
- Slow
- Monolingual
- Performance?

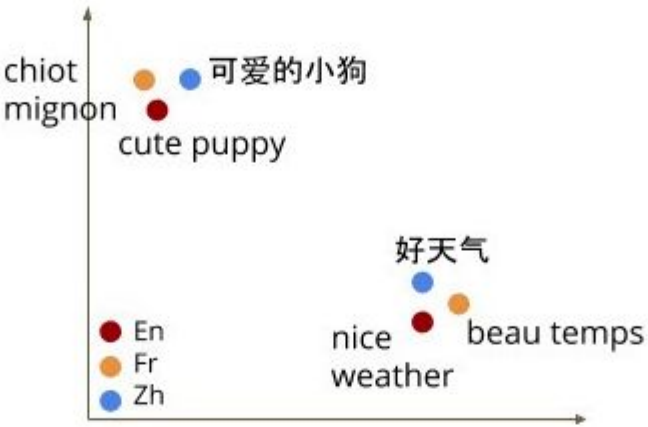




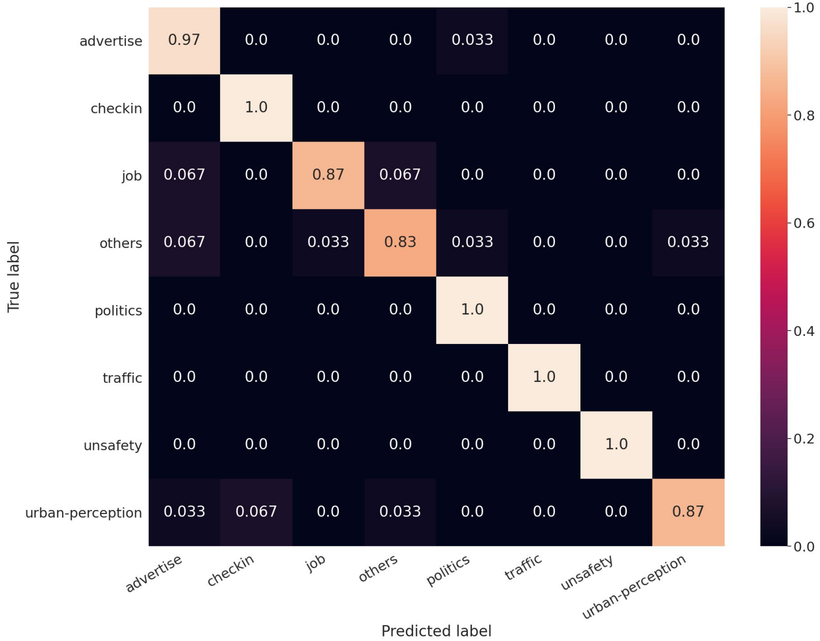


Attempt 3.2: Intent Recognition

- **LaBSE**: Language-Agnostic BERT Sentence Embedding
- Combined with a fast classifier, e.g., Logistic Regression

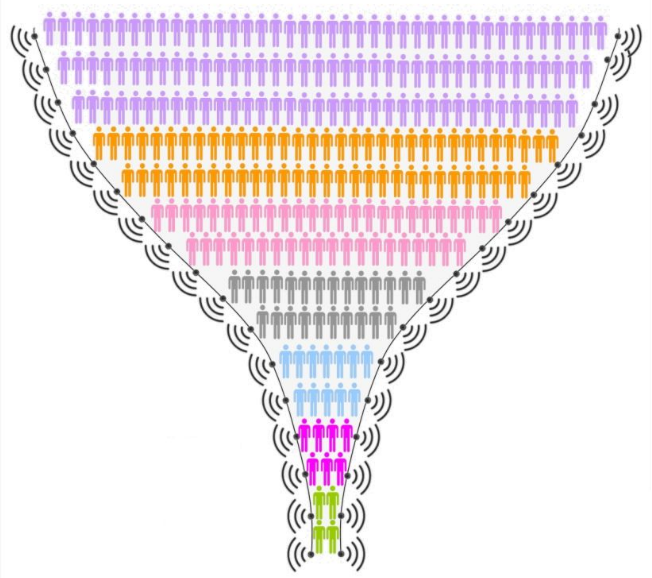


Multilingual Embedding Space



Problem 2: Only a small part of collected tweets are geotagged (~10%) and some of them, mainly those related to traffic and unsafety, are posted by news agencies

Million of tweets

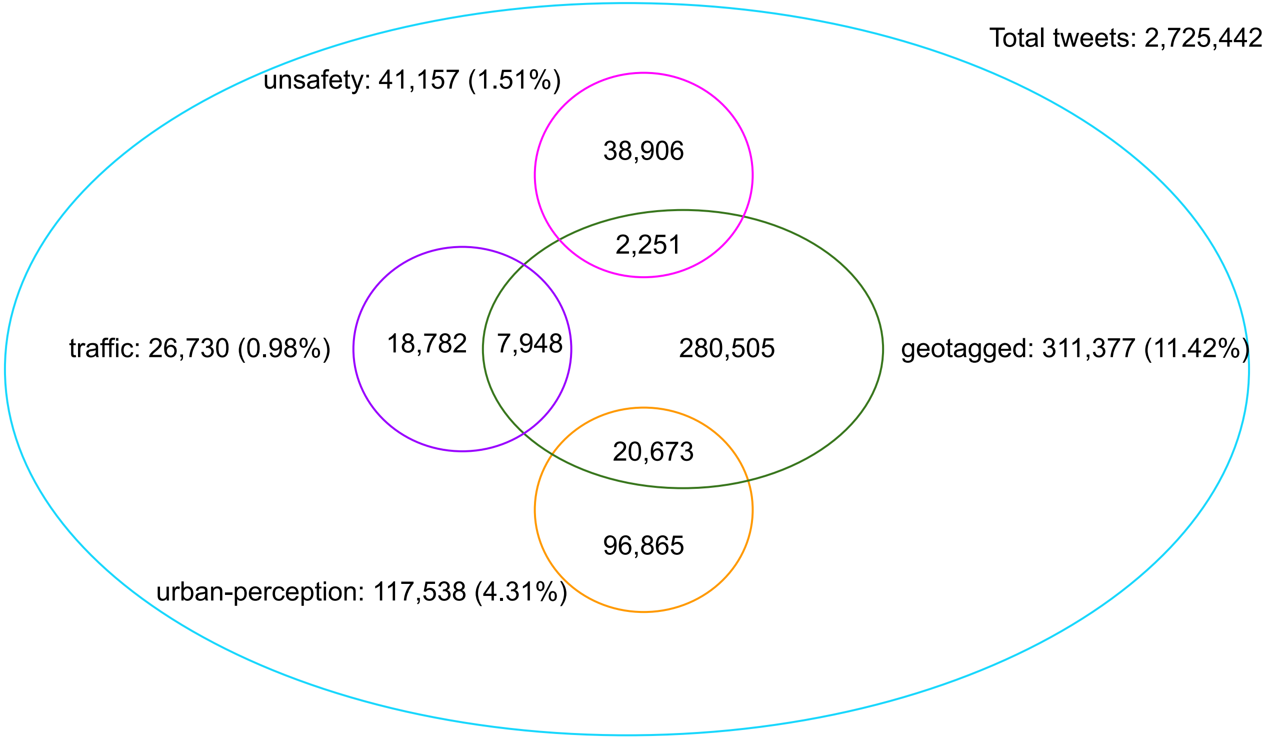


A few thousand of tweets

Hypothesis: If there is an address in the text, we can use NER to get it and applying a geocoding algorithm to obtain the most likely geolocation

Validation Dataset

- Tweets from October to December 2017 in Chicago, IL



Attempt 1: Slot Filling

Sentence:	Show	flights	from	Boston	to	New	York	today
Slots:	O	O	O	B-dept	O	B-arr	I-arr	B-date
Named Entity:				city start: 18 end: 23		city start: 28 end: 35		date start: 37 end: 41

- Manually label a data set to train a supervised model to predict address entities (POI, City, State, Country, Neighborhood, Avenue/Road/Street)
    - 1,200 samples

- Create a RNN model (BiLSTM)

Attempt 2: spaCy Library

Requirements

```
pip install spacy
python -m spacy download en_core_web_sm
```

Code:

```
import spacy

nlp = spacy.load('en_core_web_sm')

# print(spacy.explain("GPE"))

sentence = "Show flights from Boston to New York today"

doc = nlp(sentence)

for ent in doc.ents:
    print(ent.text, ent.start_char, ent.end_char, ent.label_)
```

Output

```
Boston 18 24 GPE
New York 28 36 GPE
today 37 42 DATE
```

Labels*

```
CARDINAL: Numerals that do not fall under another type
DATE: Absolute or relative dates or periods
EVENT: Named hurricanes, battles, wars, sports events,
etc
FAC: Buildings, airports, highways, bridges, etc
GPE: Countries, cities, states
LANGUAGE: Any named language
LAW: Named documents made into laws
LOC: Non-GPE locations, mountain ranges, bodies of water
MONEY: Monetary values, including unit
NORP: Nationalities or religious or political groups
ORDINAL: "first", "second", etc.
ORG: Companies, agencies, institutions, etc
PERCENT: Percentage, including "%"
PERSON: People, including fictional
PRODUCT: Objects, vehicles, foods, etc. (not services)
QUANTITY: Measurements, as of weight or distance
TIME: Times smaller than a day
WORK_OF_ART: Titles of books, songs, etc
```

Result

intent	Non-geotagged		Geotagged	
	Has entity	Has not entity	Has entity	Has not entity
traffic	**4,594**	14,188	**4,206**	3,742
unsafety	**10,039**	28,867	**1,205**	1,046
urban-p.	**18,402**	78,463	8,600	12,073

Problem 2.1: Geocoding

Attempt 1: Geocoder library (Multiple different geocoding provider such as Google, Bing, OSM & so on)

Requirements

```
pip install geocoder
## Installing Nominatim:
## http://nominatim.org/release-docs/latest/admin/Installation/
```

Code:

```
import geocoder

g = geocoder.osm(“11 Wall Street, New York”)

print(g.json)
```

Output

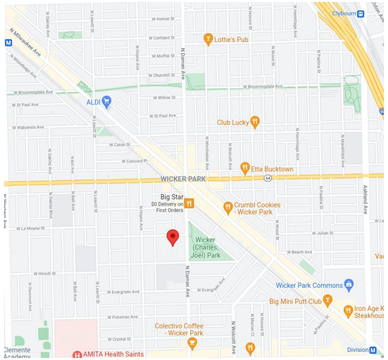
```
{
    "x": -74.010865,
    "y": 40.7071407,
    "addr:country": "United States of America",
    "addr:state": "New York",
    "addr:housenumber": "11",
    "addr:postal": "10005",
    "addr:city": "NYC",
    "addr:street": "Wall Street"
}
```

Example: urban-perception

Input

```
text = “Ending my year in Wicker Park [FAC]. The hipster part of
town where the amount of skinny jeans and man…
https://t.co/tweet_code”

fetch = “wicker park, Chicago, il”
```



Output


```
Output =
{
        'accuracy': 0.7456954929675734,
        'address': 'Wicker Park, Chicago, Cook County,
Illinois, 60622, United States',
        'bbox': {
                'northeast': [41.9178018, -87.6681551],
                'southwest': [41.8978018, -87.6881551]
        },
        'city': 'Chicago',
        'confidence': 7,
        'country': 'United States',
        'country_code': 'us',
        'county': 'Cook County',
        'importance': 0.7456954929675734,
        'lat': 41.9078018,
        'lng': -87.6781551,
        'neighborhood': 'Wicker Park',
        'ok': True,
        ...
        'quality': 'neighbourhood',
        'raw': {...}
}
```