Graph Clustering for Natural Language Processing

Lecture at ESSLLI 2022

DOI: 10.5281/zenodo.6667766

Outline

Section 1

Introduction

Introduction

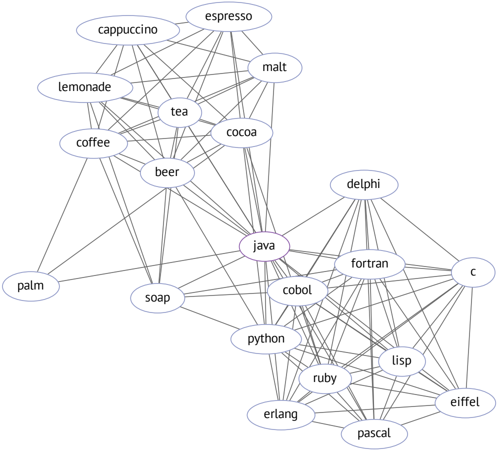
- Linguistic phenomena instantiate in linguistic data, showing interconnections and relationships
- Often, we need to learn more about the data and how these data are organized
- **Graph clustering**, as an *unsupervised learning* technique, captures the *implicit structure* of the data

Core Idea: Graphs are a Representation

After constructing it explicitly we can extract useful knowledge from it.

Motivation I

Look at this *distributional thesaurus* again!

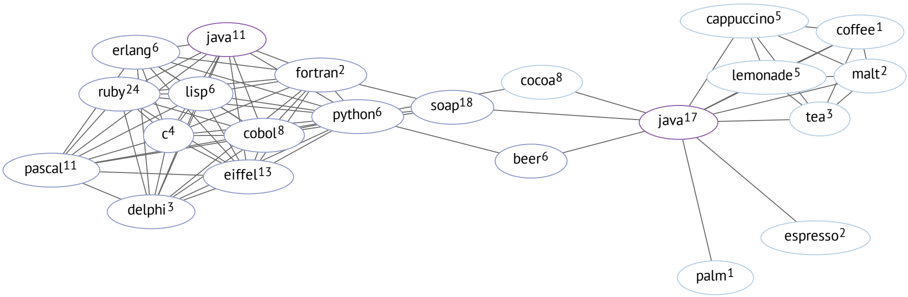


Source: Ustalov et al. (2019)

- Can we say anything interesting about the words here?
- In particular, what is interesting about the word "java"?
- Can we capture word meanings and relationships from this graph?

Motivation II

Yes, as soon as we employ the graph's structure and observe linguistic regularities.



Source: Ustalov et al. (2019)

This graph is a *disambiguated* distributional thesaurus that is obtained using graph clustering.

Successful Applications

Graph clustering helps in addressing very challenging NLP problems:

- word sense induction (Biemann, 2006)
- cross-lingual semantic relationship induction (Lewis et al., 2013)
- making sense of word embeddings (Pelevina et al., 2016)
- text summarization (Azadani et al., 2018)
- entity resolution from multiple sources (Tauer et al., 2019)

Beyond these applications, clustering is generally useful for

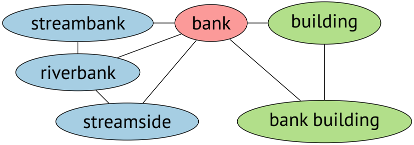
- bootstrapping the language resource
- exploring the structure of the data

Problem Formulation

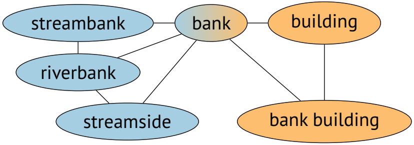
Given an *undirected* graph $G = (V, E)$, we are interested in obtaining a set cover for V called *clustering* C of this graph:

$$V = \bigcup_{C^i \in C} C^i$$

Hard Clustering



Soft Clustering



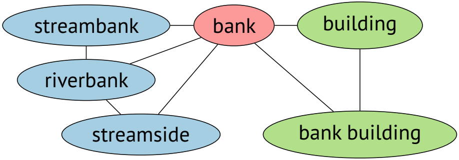
- Like in other *unsupervised learning* tasks, similar objects are expected to be close, while non-similar are not
- Every algorithm defines what good clustering is

Section 2

Hard Clustering

Hard Clustering

- **Hard clustering** algorithms (partitionings) produce non-overlapping clusters: $C^i \cap C^j = \emptyset \iff i \neq j, \forall C^i, C^j \in C$
- We will demonstrate several popular graph clustering algorithms: Spectral Clustering, Chinese Whispers, Markov Clustering, and Louvain
- There are *a lot* of other clustering algorithms!



Spectral Clustering

- **Spectral Clustering** performs an embedding of the Laplacian matrix and then applies a clustering algorithm (von Luxburg, 2007)
- Laplacians are used as they are symmetric and have $|V|$ non-negative eigenvalues
- We will focus on the algorithm by Ng et al. (2002) that uses a normalized Laplacian L^{norm} and k-Means (Hartigan et al., 1979)

$$L^{\mathrm{norm}} = U \Lambda U^{-1}$$

Columns of U are *orthogonal* eigenvectors of L^{norm} and Λ is a diagonal matrix of its eigenvalues.

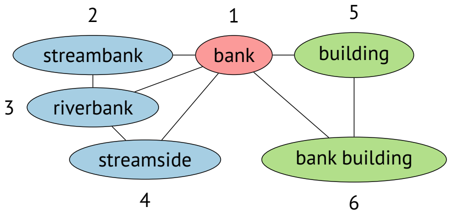
Spectral Clustering: Algorithm

Input: graph $G = (V, E)$, adjacency matrix A, degree matrix D, number of clusters k

Output: clustering C

1: $L^{\text{norm}} \leftarrow D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$
2: $U\Lambda U^{-1} \leftarrow \texttt{ein}(L^{\text{norm}})$ ▷ Assume the eigenvalues are descending
3: $T_{ij} \leftarrow \frac{U_{ij}}{\sqrt{\sum_{1 \le l \le k} U_{il}^2}}$ **for all** $1 \le i \le |V|, 1 \le j \le k$
4: $C \leftarrow \texttt{k-means}(T, k)$ ▷ $|V|$ objects and k clusters
5: **return** C

Spectral Clustering: Example



$$T = \begin{pmatrix} .53 & 0 & .85 \\ -.99 & 0 & .13 \\ .62 & 0 & -.78 \\ -.99 & 0 & .13 \\ -.16 & -.93 & -.33 \\ -.16 & .93 & -.33 \end{pmatrix}$$

This is an example using the graph from Ustalov et al. (2019, Figure 2)

Spectral Clustering: Discussion

Pros:

- \+ Sound method that optimizes the normalized cut (Shi et al., 2000)
- \+ Handles very complex clusters

Cons:

- − Need to specify k and the clustering algorithm
- − Computationally expensive

Implementations:

- https://github.com/scikit-learn/scikit-learn
- https://github.com/nlpub/watset-java

A great tutorial on spectral clustering
is available in von Luxburg (2007).

Chinese Whispers (CW)

- **Chinese Whispers** (CW) is a *randomized* hard clustering algorithm for both weighted and unweighted graphs (Biemann, 2006)
- Named after a famous children's game, it uses random shuffling to induce clusters
- Originally designed for such NLP tasks as word sense induction, language separation, etc.

Source: Adamovich (2015)

Chinese Whispers: Algorithm

Input: graph $G = (V, E), \text{weight} : (G_u, i) \to \mathbb{R}, \forall u \in V, 1 \leq i \leq |V|$
Output: clustering C

1: $\text{label}(V_i) \leftarrow i$ **for all** $1 \leq i \leq |V|$ ▷ Initialization
2: **while** labels change **do** ▷ $\text{labels}(G_u)$ is a set of node labels in G_u
3: **for all** $u \in V$ in random order **do**
4: $\text{label}(u) \leftarrow \arg\max_{i \in \text{labels}(G_u)} \text{weight}(G_u, i)$ ▷ Pick the most weighted label in G_u
5: $C \leftarrow \{\{u \in V : \text{label}(u) = i\} : i \in \text{labels}(G)\}$
6: **return** C

Chinese Whispers: Label Weighting

Typical strategies to weigh the labels in the neighborhood G_u of u in G:

- Sum of the edge weights corresponding to the label i (`top`):

$$\text{weight}(G_u, i) = \sum_{\{u,v\} \in E_u : \text{label}(v) = i} w(u, v)$$

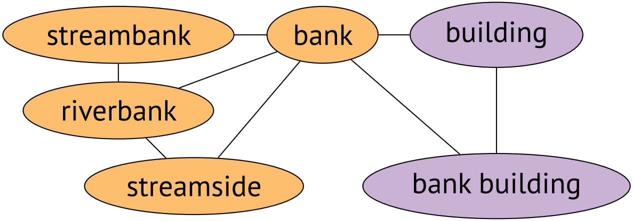
- Use the node degree $\deg(v)$ to amortize highly-weighted edges (`linear`):

$$\text{weight}(G_u, i) = \sum_{\{u,v\} \in E_u : \text{label}(v) = i} \frac{w(u,v)}{\deg(v)}$$

- Use log-degree for amortization (`log`):

$$\text{weight}(G_u, i) = \sum_{\{u,v\} \in E_u : \text{label}(v) = i} \frac{w(u,v)}{\log(1 + \deg(v))}$$

Chinese Whispers: Example



This is an example using the graph from Ustalov et al. (2019, Figure 2)

Chinese Whispers: Discussion

Pros:

- \+ Very simple and non-parametric
- \+ Very fast, the running time is $O(|E|)$
- \+ Works well for a lot of NLP tasks

Cons:

- − Every run yields different results
- − Node oscillation is possible
- − No convergence guarantee

Implementations:

- https://github.com/uhh-lt/chinese-whispers
- https://github.com/nlpub/chinese-whispers-python

Markov Clustering (MCL)

- **Markov Clustering** (MCL) is a *stochastic* hard clustering algorithm that simulates *flows* in a graph using **random walks** (van Dongen, 2000)
- The algorithm makes a series of adjacency matrix transformations to obtain the partitioning: *expansion* and *inflation*
- MCL has been applied in a number of different domains, mostly in bioinformatics (Vlasblom et al., 2009)
- Similar to Affinity Propagation (Frey et al., 2007)

Source: Merrill (2014)

Markov Clustering: Algorithm

Input: graph $G = (V, E)$, adjacency matrix A,
expansion parameter $e \in \mathbb{N}$, inflation parameter $r \in \mathbb{R}^+$

Output: clustering C

```
A_ii ← 1 for all 1 ≤ i ≤ |V|                                   ▷ Add self-loops
A_ij ← A_ij / Σ_{1≤k≤|V|} A_kj for all 1 ≤ i ≤ |V|, 1 ≤ j ≤ |V|   ▷ Normalize
while A changes do
    A ← A^e                                                      ▷ Expand
    A_ij ← A_ij^r for all 1 ≤ i ≤ |V|, 1 ≤ j ≤ |V|                ▷ Inflate
    A_ij ← A_ij / Σ_{1≤k≤|V|} A_kj for all 1 ≤ i ≤ |V|, 1 ≤ j ≤ |V|   ▷ Normalize
C ← {{V_j ∈ V : A_ij ≠ 0} : 1 ≤ i ≤ |V|, 1 ≤ j ≤ |V|}
return C
```

$A_{ii} \leftarrow 1$ **for all** $1 \leq i \leq |V|$ ▷ Add self-loops

$A_{ij} \leftarrow \frac{A_{ij}}{\sum_{1 \leq k \leq |V|} A_{kj}}$ **for all** $1 \leq i \leq |V|, 1 \leq j \leq |V|$ ▷ Normalize

while A changes **do**

  $A \leftarrow A^e$ ▷ Expand

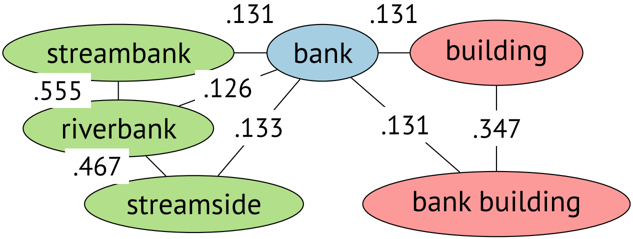
  $A_{ij} \leftarrow A_{ij}^r$ **for all** $1 \leq i \leq |V|, 1 \leq j \leq |V|$ ▷ Inflate

  $A_{ij} \leftarrow \frac{A_{ij}}{\sum_{1 \leq k \leq |V|} A_{kj}}$ **for all** $1 \leq i \leq |V|, 1 \leq j \leq |V|$ ▷ Normalize

$C \leftarrow \{\{V_j \in V : A_{ij} \neq 0\} : 1 \leq i \leq |V|, 1 \leq j \leq |V|\}$

return C

Markov Clustering: Example



- This is an example using the graph from Ustalov et al. (2019, Figure 2)

Markov Clustering: Discussion

Pros:

- + Eventually, the algorithm converges (but there is no formal proof)
- + Works well for a lot of NLP tasks

Cons:

- − A naïve implementation is slow; the worst-case running time is $O(|V|^3)$, yet pruning allows achieving $O(|V|k)$, where k is the number of resources per node

Implementations:

https://micans.org/mcl/

- Expansion e makes farther nodes reachable
- Inflation r changes the granularity of the clusters

Modularity

Modularity measures the density of connections inside clusters vs. the density of those between clusters (Blondel et al., 2008).

Let $m = \frac{1}{2}\sum_{ij} A_{ij}$ be the number of edges in a graph $G = (V, E)$, $k_i = \deg(u_i)$ be the degree of node $u \in V$, and $\delta(c_i, c_j) = 1$ if $c_i = c_j$ and 0 otherwise.

Newman (2004) defines the modularity $Q \in [-\frac{1}{2}, 1]$ as

$$Q = \frac{1}{2m}\sum_{ij}\left[A_{ij} - \frac{k_i k_j}{2m}\delta(c_i, c_j)\right]$$

Graphs with high modularity have dense communities of nodes.

Louvain Method

- Blondel et al. (2008) proposed the algorithm called *Louvain* that maximizes the modularity of a graph
- Louvain method achieves modularity gains by moving an isolated node $u_i \in V$ into a cluster $C^j \subseteq V$:

$$\Delta Q = \left[\frac{\Sigma_{\text{in}} + k_{i,\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}} + k_i}{2m}\right)^2\right] - \left[\frac{\Sigma_{\text{in}}}{2m} - \left(\frac{\Sigma_{\text{tot}}}{2m}\right)^2 - \left(\frac{k_i}{2m}\right)^2\right],$$

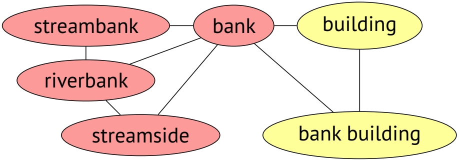
where m is the number of edges, $k_i = \deg(u_i)$ is the degree of $u \in V$, Σ_{in} is the sum of edge weights inside C^j, Σ_{tot} is the sum of weights of the edges incident to nodes in C^j, and $k_{i,\text{in}}$ is the sum of edge weights from u_i to nodes in C^j

Louvain Method: Pseudocode (Simplified)

Input: graph $G = (V, E)$
Output: clustering C

1: **while** Q increases **do**
2: $C^i \leftarrow \{u_i\}$ **for all** $1 \leq i \leq |V|$ ▷ Singleton clusters
3: **while** clusters are changed **do**
4: **for all** $u_i \in V$ **do** ▷ Often randomized
5: $j \leftarrow \underset{\substack{1 \leq j \leq |C|: \\ C^j \subseteq V(u_i) \cup \{u_i\}}}{\arg\max} \Delta Q$ ▷ Find the maximizing cluster C^j
6: $C^j \leftarrow C^j \cup \{u_i\}$ ▷ Add the node to C^j
7: $C^i \leftarrow C^i \setminus \{u_i\}$ ▷ Remove the node from C^i
8: **return** clusters of G

Louvain Method: Example



$$Q = 0.16015625$$

- This is an example using the graph from Ustalov et al. (2019, Figure 2)

Louvain Method: Discussion

Pros:

- \+ The algorithm is non-parametric
- \+ Sound method that performs modularity maximization
- \+ Fast, the empirical running time is $O(|V| \log(|V|))$
- \+ Hierarchical clustering can be obtained “for free”

Cons:

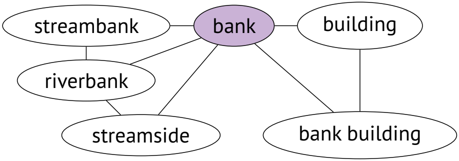
- − Modularity is not sensitive enough to detect small communities
- − Q lacks a clear global optimum (Good et al., 2010)

Implementations:

- https://sourceforge.net/projects/louvain/
- https://networkx.org/ (Hagberg et al., 2008)
- https://gephi.org/

Hard Clustering: Wrap-Up

- Hard clustering algorithms allow partitioning the graph
- OK, but how about the fact that the word "bank" is polysemous?
- These algorithms will treat this word incorrectly
- Is there a way for addressing this issue?

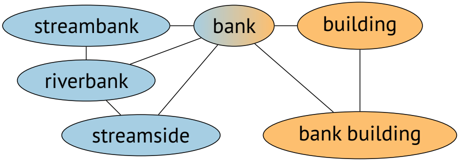


Section 3

Soft Clustering

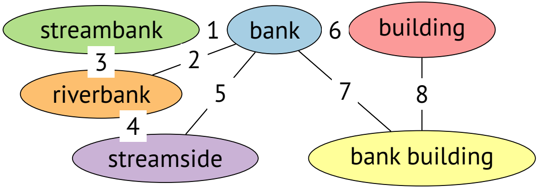
Soft Clustering

- **Soft clustering** algorithms permit cluster overlapping, i.e., a node can be a member of several clusters: $|C^i \in C : u \in C^i| \geq 1, \forall u \in V$
- A *harder* problem as the problem space is larger
- We will demonstrate two different soft clustering algorithms: MaxMax and Watset



Line Graphs I

- Although the nodes of this graph G may have multiple meanings, each edge connects only one meaning with another

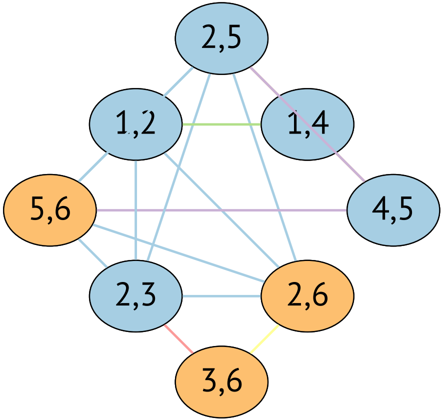


- What if we build a representation that uses this invariant and then employ it to induce the meanings?
- This representation is well-known: a **line graph** $L(G)$
- We can just build it, cluster it, and then retrieve the original nodes from clusters

- The original graph G had 6 nodes and 8 edges, and the line graph $L(G)$ has 8 nodes and 17 edges
- This operation becomes computationally very expensive if the graph is large:

$$|E_{L(G)}| = \sum_{u \in V} \binom{\deg(u)}{2}$$

- Although it does what we want, we need more scalable approaches



MaxMax

- **MaxMax** is a *soft* clustering algorithm designed for *weighted* graphs, such as co-occurrence graphs (Hope et al., 2013a)
- MaxMax transforms the input undirected weighted graph G into an unweighted directed graph G'
- Then, it extracts *quasi-strongly connected* subgraphs from G', which are overlapping clusters

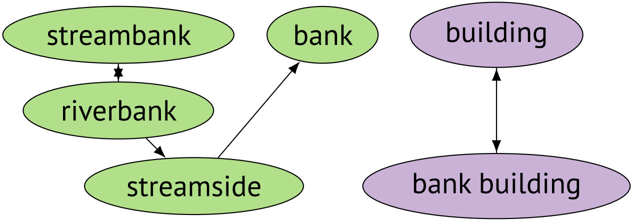
Source: Rahman Rony (2016)

MaxMax: Algorithm

Input: graph $G = (V, E)$, weighting function $w : E \to \mathbb{R}$
Output: clustering C

$E' \leftarrow \emptyset$
for all $\{u, v\} \in E$ **do**
  if $w(u, v) = \max_{v' \in V_u} w(u, v')$ **then**
    $E' \leftarrow E' \cup \{(v, u)\}$
$G' = (V, E')$
$\mathrm{root}(u) \leftarrow \mathrm{true}$ **for all** $u \in V$
for all $u \in V$ **do**
  if $\mathrm{root}(u)$ **then**
    for all $v \in \mathrm{succ}(u)$ **do** ▷ Successors of u in G'
      $\mathrm{root}(u) \leftarrow \mathrm{false}$
$C \leftarrow \{\{u\} \cup \mathrm{succ}(u) : u \in V, \mathrm{root}(u)\}$
return C

MaxMax: Example



This is an example using the graph from Ustalov et al. (2019, Figure 2)

MaxMax: Discussion

Pros:

- The algorithm is non-parametric
- Very fast, the running time is $O(|E|)$, like CW
- Works well for word sense induction (Hope et al., 2013b)

Cons:

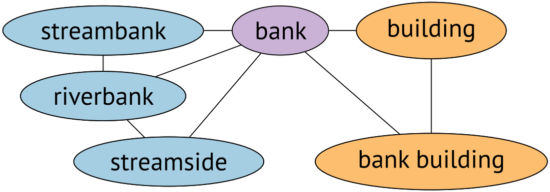
- Applicability seems to be limited (Ustalov et al., 2019)
- A weighted graph is required

Implementations:

- https://github.com/nlpub/watset-java

Graph-Based Word Sense Induction (WSI)

- Dorow et al. (2003) proposed a nice approach for **word sense induction** (WSI) using graphs
- Extract the *node neighborhood*, remove the node, and cluster the remaining graph
- Every cluster C^i corresponds to the *context* of the i-th sense of the node



Source: Kittner (2015)

Watset

- **Watset** is not a clustering algorithm
- However, it is a *meta-algorithm* for turning *hard* clustering algorithms into *soft* clustering algorithms
- Watset **transforms** the input graph by replacing each node with one or more *senses* of this node using *word sense induction* (Dorow et al., 2003) and *context disambiguation* (Faralli et al., 2016)
- We will focus on the better variation called **Simplified Watset** (or Watset§) as described in Ustalov et al. (2019, Section 3.4)

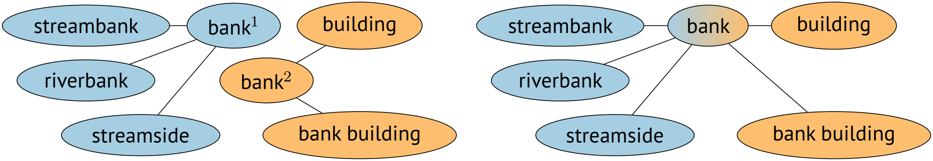
Source: FreePhotosART (2016)

Watset: Algorithm

Input: graph $G = (V, E)$, algorithms $\mathrm{Cluster}_{\textsf{Local}}$ and $\mathrm{Cluster}_{\textsf{Global}}$
Output: clusters C

for all $u \in V$ **do** ▷ Local Step
  $V_u \leftarrow \{v \in V : \{u, v\} \in E\}$ ▷ Note that $u \notin V_u$
  $E_u \leftarrow \{\{v, w\} \in E : v, w \in V_u\}$
  $G_u \leftarrow (V_u, E_u)$
  $C_u \leftarrow \mathrm{Cluster}_{\textsf{Local}}(G_u)$ ▷ Cluster the open neighborhood of u
  for all $C_u^i \in C_u$ **do**
    for all $v \in C_u^i$ **do**
      $\mathrm{senses}[u][v] \leftarrow i$ ▷ Node v is connected to the i-th sense of u
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{u^i\}$
$\mathcal{E} \leftarrow \{\{u^{\mathrm{senses}[u][v]}, v^{\mathrm{senses}[v][u]}\} \in \mathcal{V}^2 : \{u, v\} \in E\}$ ▷ Global Step
$\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E})$
$\mathcal{C} \leftarrow \mathrm{Cluster}_{\textsf{Global}}(\mathcal{G})$ ▷ Prepare to remove node labels
return $\{\{u \in V : \hat{u} \in \mathcal{C}^i\} \subseteq V : \mathcal{C}^i \in \mathcal{C}\}$

Watset: Example



- This is an example from Ustalov et al. (2019)

Watset: Discussion

Pros:

- Conceptually very simple
- Scales very well
- Shows very good results on very different tasks (Ustalov et al., 2019)

Cons:

- Adds overhead for local clustering of $O(|V|^2\Delta(G)^2)$ for CW and $O(|V|^3\Delta(G)^3)$ for MCL
- Good as long as the underlying clustering algorithms are good

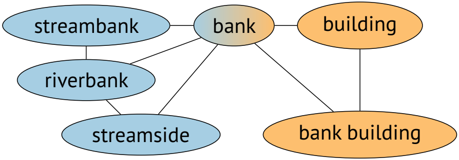
Implementations:

- https://github.com/nlpub/watset-java
- https://github.com/dustalov/watset

The Java implementation of Watset also contains CW, MCL, and MaxMax. **Feel free to play with them!**

Soft Clustering: Wrap-Up

- Soft clustering handles polysemous words and other kinds of multiple presences of nodes in the clusters
- Be careful with the assumptions the algorithms make and the transformations they perform



Section 4

Case Studies

Case Studies

- **Synset Induction** from Synonymy Dictionaries (Ustalov et al., 2019, Section 4)
- Unsupervised Semantic **Frame Induction** (Ustalov et al., 2019, Section 5)
- Making Sense of Word Embeddings (Pelevina et al., 2016)

Source: Finnsson (2017)

Synset Induction

- Ontologies and thesauri are crucial to many NLP applications that require common sense reasoning
- The building blocks of WordNet (Fellbaum, 1998) are **synsets**, sets of mutual synonyms

  {broadcast, program, programme}
- Can we build synsets from scratch using just *synonymy dictionaries* like Wiktionary?

Source: Buissinne (2016)

Synset Induction: WordNet

WordNet Search - 3.1
- WordNet home page - Glossary - Help

Word to search for: cat [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: (gloss) "an example sentence"

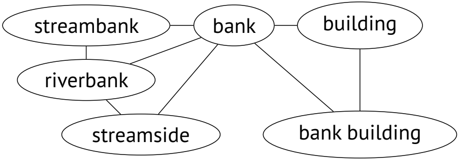
Noun

- S: (n) **cat**, true cat (feline mammal usually having thick soft fur and no ability to roar: domestic cats; wildcats)
  - ***direct hyponym*** / *full hyponym*
    - S: (n) domestic cat, house cat, Felis domesticus, Felis catus (any domesticated member of the genus Felis)
    - S: (n) wildcat (any small or medium-sized cat resembling the domestic cat and living in the wild)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - S: (n) feline, felid (any of various lithe-bodied roundheaded fissiped mammals, many with retractile claws)
      - S: (n) carnivore (a terrestrial or aquatic flesh-eating mammal) *"terrestrial carnivores have four or five clawed digits on each limb"*
        - S: (n) placental, placental mammal, eutherian, eutherian mammal (mammals having a placenta; all mammals except monotremes and marsupials)
          - S: (n) mammal, mammalian (any warm-blooded vertebrate having the skin more or less covered with hair; young are born alive except for the small subclass of monotremes and nourished with milk)
            - S: (n) vertebrate, craniate (animals having a bony or cartilaginous skeleton with a segmented spinal column and a large brain enclosed in a skull or cranium)
              - S: (n) chordate (any animal of the phylum Chordata having a notochord or spinal column)
                - S: (n) animal, animate being, beast, brute, creature, fauna (a living organism characterized by voluntary movement)
                  - S: (n) organism, being (a living thing that has (or can develop) the ability to act or function independently)
                    - S: (n) living thing, animate thing (a living (or once living) entity)
                      - S: (n) whole, unit (an assemblage of parts that is regarded as a single entity) *"how big is that part compared to the whole?"; "the team is a unit"*
                        - S: (n) object, physical object (a tangible and visible entity; an entity that can cast a shadow) *"it was full of rackets, balls and other objects"*
                          - S: (n) physical entity (an entity that has physical existence)
                            - S: (n) entity (that which is perceived or known or inferred to have its own distinct existence (living or nonliving))

Source: `http://wordnetweb.princeton.edu/perl/webwn`

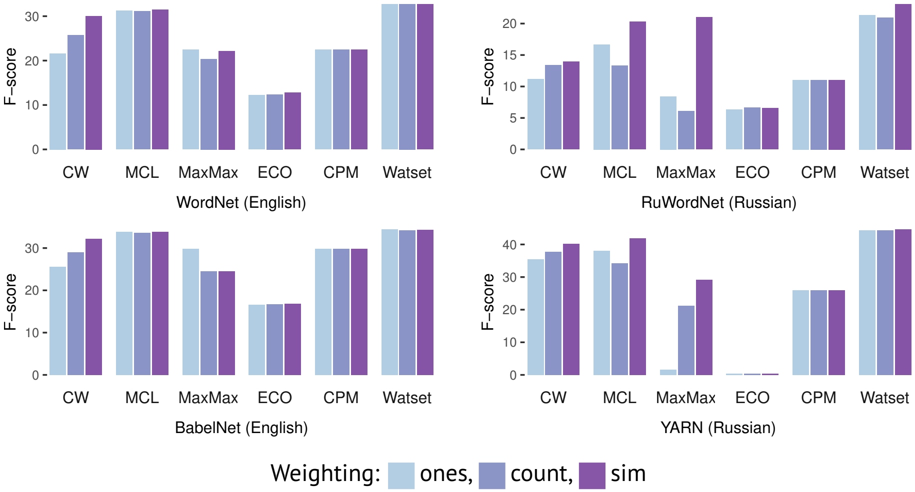
Synset Induction: Approach

1. Construct a weighted undirected graph using synonymy pairs from Wiktionary as edges
2. Weight them using cosine similarity between the corresponding word embeddings
3. Cluster this graph and treat the clusters as synsets



Code and Data: https://github.com/dustalov/watset

Watset showed the best results according to paired F_1-score.



Source: Ustalov et al. (2019)

Synset Induction: Example

Size	Synset
2	decimal point, dot
2	wall socket, power point
3	gullet, throat, food pipe
3	CAT, computed axial tomography, CT
4	microwave meal, ready meal, TV dinner, frozen dinner
4	mock strawberry, false strawberry, gurbir, Indian strawberry
5	objective case, accusative case, oblique case, object case, accusative
5	discipline, sphere, area, domain, sector
6	radio theater, dramatized audiobook, audio theater, radio play, radio drama, audio play
6	integrator, reconciler, consolidator, mediator, harmonizer, uniter
7	invite, motivate, entreat, ask for, incentify, ask out, encourage
7	curtail, craw, yield, riding crop, harvest, crop, hunting crop

Source: Ustalov et al. (2019)

Frame Induction

- A **semantic frame** is a collection of facts that specify features, attributes, and functions (Fillmore, 1982)

FrameNet	**Role**	**Lexical Units (LU)**
Perpetrator	Subject	kidnapper, alien, militant
FEE	Verb	snatch, kidnap, abduct
Victim	Object	son, people, soldier, child

- Used in question answering, textual entailment, event-based predictions of stock markets, etc.
- Can we build frames from scratch using just *subject-verb-object* (SVO) triples like DepCC (Panchenko et al., 2018)?

Source: rawpixel (2017)

Kidnapping

Definition:

The words in this frame describe situations in which a Perpetrator carries off and holds the Victim against his or her will by force.

Two men KIDNAPPED a Millwall soccer club employee, police said last night.

Not even the ABDUCTION of his children by Captain Hook and his scurvy sidekick, Smee, can shake Peter's scepticism.

FEs:

Core:

Perpetrator [Perp] **Semantic Type:** Sentient	The Perpetrator is the person (or other agent) who carries off and holds the Victim against his or her will.
Victim [Vict] **Semantic Type:** Sentient	The Victim is the person who is carried off and held against his/her will.

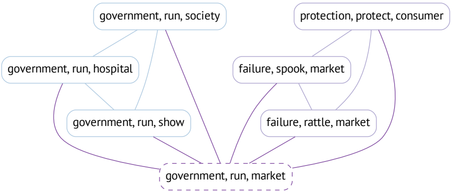
Lexical Units:

abduct.v, abducted.a, abduction.n, abductor.n, kidnap.v, kidnapped.a, kidnapper.n, kidnapping.n, nab.v, shanghai.v, snatch.v, snatcher.n

Source: https://framenet.icsi.berkeley.edu/fndrupal/luIndex

Frame Induction: Approach

1. Use word embeddings to embed each triple $t = (s, v, o)$ in a low-dimensional *vector space* as $\vec{t} = \vec{s} \parallel \vec{v} \parallel \vec{o}$
2. Construct a weighted undirected graph using $k \in \mathbb{N}$ nearest neighbors of each triple vector
3. Cluster this graph and extract *triframes* by aggregating the corresponding roles

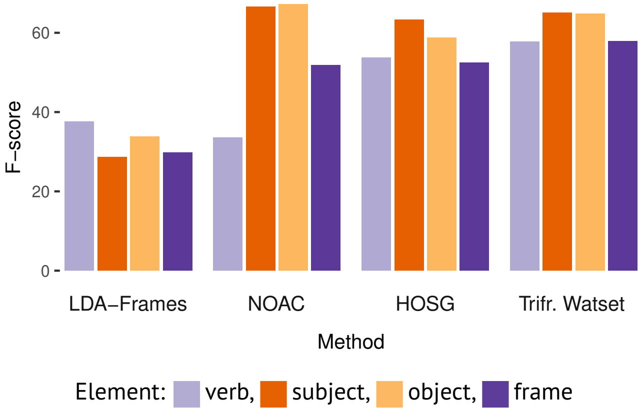


Source: Ustalov et al. (2019)

Code and Data: `https://github.com/uhh-lt/triframes`

Frame Induction: Results

Triframes outperformed state-of-the-art frame induction approaches, including Higher-Order Skip-Gram (HOSG) and LDA-Frames, on the FrameNet corpus (Baker et al., 1998) as according to F_1 (nmPU / niPU).



Source: Ustalov et al. (2019)

Frame Induction: Good Examples

Subjects: expert, scientist, lecturer, engineer, analyst
Verbs: study, examine, tell, detect, investigate, do, observe, hold, find, have, predict, claim, notice, give, discover, explore, learn, monitor, check, recognize, demand, look, call, engage, spot, inspect, ask
Objects: view, problem, gas, area, change, market

Subjects: leader, officer, khan, president, government, member, minister, chief, chairman
Verbs: belong, run, head, spearhead, lead
Objects: party, people

Subjects: evidence, research, report, survey
Verbs: prove, reveal, tell, show, suggest, confirm, indicate, demonstrate
Objects: method, evidence

Source: Ustalov et al. (2019)

Frame Induction: Bad Examples

Subjects: wine, act, power
Verbs: hearten, bring, discourage, encumber, ...*432 more verbs*..., build, chew, unsettle, snap
Objects: right, good, school, there, thousand

Subjects: parent, scientist, officer, event
Verbs: promise, pledge
Objects: parent, be, good, government, client, minister, people, coach

Subjects: people, doctor
Verbs: spell, steal, tell, say, know
Objects: egg, food, potato

Source: Ustalov et al. (2019)

Making Sense of Word Embeddings

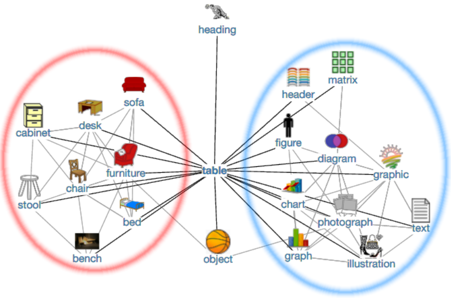
Such word embedding models as Word2Vec (Mikolov et al., 2013) capture linguistic regularities but do not take into account individual *word senses*.

- $\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Russia}} \approx \overrightarrow{\text{Moscow}}$
- $\overrightarrow{\text{apple}} - \overrightarrow{\text{apples}} \approx \overrightarrow{\text{car}} - \overrightarrow{\text{cars}}$

Pelevina et al. (2016) proposed **SenseGram**, a word sense embedding approach that uses simple arithmetical operations on word embeddings.

Making Sense of Word Embeddings: Approach

1. Build a co-occurrence graph and perform node sense induction
2. Retrieve word embeddings for each word in each cluster
3. Average word embeddings in each cluster
4. Treat the averaged vectors as sense embeddings



Source: Pelevina et al. (2016)

Code and Data: https://github.com/uhh-lt/sensegram

Making Sense of Word Embeddings: Example

Vector	Nearest Neighbours
table	tray, bottom, diagram, bucket, brackets, stack, basket, list, parenthesis, cup, trays, pile, playfield, bracket, pot, drop-down, cue, plate
table^0	leftmost^0, column^1, randomly^0, tableau^1, top-left^0, indent^1, bracket^3, pointer^0, footer^1, cursor^1, diagram^0, grid^0
table^1	pile^1, stool^1, tray^0, basket^0, bowl^1, bucket^0, box^0, cage^0, saucer^3, mirror^1, birdcage^0, hole^0, pan^1, lid^0

Source: Pelevina et al. (2016)

Making Sense of Word Embeddings: Results

- Such a simple approach shows comparable results to more sophisticated methods, e.g., on SemEval-2013 Task 13 (Jurgens et al., 2013)
- Obtained vectors can be used as baselines or features in downstream applications

Model	**WNDCG**	**FB-Cubed**
Most Frequent Sense	0.302	**0.631**
AI-KU (remove5-add1000)	0.330	0.463
UoS (top-3)	0.370	0.451
La Sapienza (2)	**0.394**	–
AdaGram (100-d), $\alpha = 0.05$	0.318	0.470
SenseGram Word2Vec Nouns	0.304	0.623

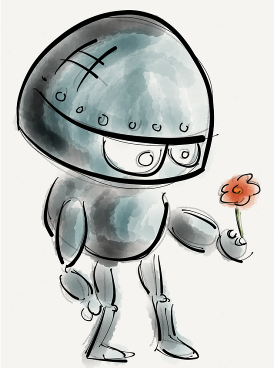
Source: Pelevina et al. (2016)

Section 5

Conclusion

Conclusion

- A graph is a meaningful representation; clustering captures its implicit structure as exhibited by data
- Clustering is useful in exploring and bootstrapping datasets
- The algorithms are well-developed and ready to use as soon as a graph is constructed
- Not covered here: algorithms for community detection from network science (Fortunato, 2010), combinatorial optimization (Peng et al., 2021)



Source: bamenny (2016)

Which Algorithm to Choose?

? Do you need *hard* clustering of a relatively small graph?

! Markov Clustering

? Do you still need *hard* clustering, but your graph is big?

! Chinese Whispers or Louvain

? Do you need *soft* clustering?

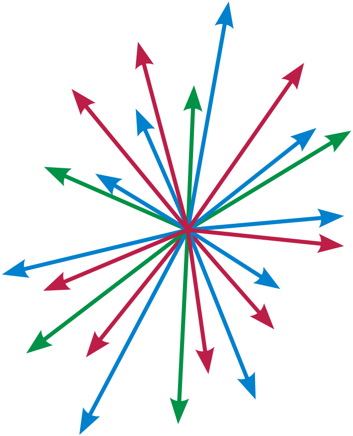
! Watset

...but My Objects are Just Vectors!

It is possible to represent the objects in a vector space as a graph (von Luxburg, 2007):

- use the k nearest neighbors,
- use all the neighbors within the ε-radius,
- use a fully-connected *weighted* graph

Think of a graph as a *discretized* vector space.



Source: Alexandrov (2007)

Questions?

Contacts

Dr. **Dmitry Ustalov**

https://github.com/dustalov

mailto:dmitry.ustalov@gmail.com

0000-0002-9979-2188

Revision: 5d35748

References I

Azadani M. N., Ghadiri N., and Davoodijam E. (2018). Graph-based biomedical text summarization: An itemset mining and sentence clustering approach. *Journal of Biomedical Informatics*, vol. 84, pp. 42–58. DOI: 10.1016/j.jbi.2018.06.005.

Baker C. F., Fillmore C. J., and Lowe J. B. (1998). The Berkeley FrameNet Project. *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. ACL '98/COLING '98. Montréal, QC, Canada: Association for Computational Linguistics, pp. 86–90. DOI: 10.3115/980845.980860.

Biemann C. (2006). Chinese Whispers: An Efficient Graph Clustering Algorithm and Its Application to Natural Language Processing Problems. *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*. TextGraphs-1. New York, NY, USA: Association for Computational Linguistics, pp. 73–80. DOI: 10.3115/1654758.1654774.

Blondel V. D. et al. (2008). Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 10, P10008. DOI: 10.1088/1742-5468/2008/10/P10008.

van Dongen S. (2000). Graph Clustering by Flow Simulation. PhD thesis. Utrecht, The Netherlands: University of Utrecht. HDL: 1874/848.

Dorow B. and Widdows D. (2003). Discovering Corpus-Specific Word Senses. *Proceedings of the Tenth Conference on European Chapter of the Association for Computational Linguistics - Volume 2*. EACL '03. Budapest, Hungary: Association for Computational Linguistics, pp. 79–82. DOI: 10.3115/1067737.1067753.

Faralli S. et al. (2016). Linked Disambiguated Distributional Semantic Networks. *The Semantic Web – ISWC 2016, 15th International Semantic Web Conference, Kobe, Japan, October 17–21, 2016, Proceedings, Part II*. Vol. 9982. Lecture Notes in Computer Science. Cham, Switzerland: Springer International Publishing, pp. 56–64. DOI: 10.1007/978-3-319-46547-0_7.

Fellbaum C. (1998). WordNet: An Electronic Database. Massachusetts, MA, USA: MIT Press. ISBN: 978-0-262-06197-1. DOI: 10.7551/mitpress/7287.001.0001.

Fillmore C. J. (1982). Frame Semantics. *Linguistics in the Morning Calm*. Seoul, South Korea: Hanshin Publishing Co., pp. 111–137.

Fortunato S. (2010). Community detection in graphs. *Physics Reports*, vol. 486, no. 3, pp. 75–174. DOI: 10.1016/j.physrep.2009.11.002.

Frey B. J. and Dueck D. (2007). Clustering by Passing Messages Between Data Points. *Science*, vol. 315, no. 5814, pp. 972–976. DOI: 10.1126/science.1136800.

Good B. H., de Montjoye Y.-A., and Clauset A. (2010). Performance of modularity maximization in practical contexts. *Physical Review E*, vol. 81, no. 4, p. 046106. DOI: 10.1103/PhysRevE.81.046106.

Hagberg A. A., Schult D. A., and Swart P. J. (2008). Exploring Network Structure, Dynamics, and Function using NetworkX. *Proceedings of the 7th Python in Science Conference*. SciPy2008. Pasadena, CA, USA, pp. 11–15. URL: https://conference.scipy.org/proceedings/SciPy2008/paper_2.

Hartigan J. A. and Wong M. A. (1979). Algorithm AS 136: A K-Means Clustering Algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108. DOI: 10.2307/2346830.

References II

Hope D. and Keller B. (2013a). MaxMax: A Graph-Based Soft Clustering Algorithm Applied to Word Sense Induction. *Computational Linguistics and Intelligent Text Processing, 14th International Conference, CICLing 2013, Samos, Greece, March 24-30, 2013, Proceedings, Part I*. Vol. 7816. Lecture Notes in Computer Science. Berlin and Heidelberg, Germany: Springer Berlin Heidelberg, pp. 368–381. DOI: 10.1007/978-3-642-37247-6_30.

Hope D. and Keller B. (2013b). UoS: A Graph-Based System for Graded Word Sense Induction. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, GA, USA: Association for Computational Linguistics, pp. 689–694. URL: https://aclanthology.org/S13-2113.

Jurgens D. and Klapaftis I. (2013). SemEval-2013 Task 13: Word Sense Induction for Graded and Non-Graded Senses. *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Atlanta, GA, USA: Association for Computational Linguistics, pp. 290–299. URL: https://aclanthology.org/S13-2049.

Lewis M. and Steedman M. (2013). Unsupervised Induction of Cross-Lingual Semantic Relations. *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. EMNLP 2013. Seattle, WA, USA: Association for Computational Linguistics, pp. 681–692. URL: https://aclanthology.org/D13-1064.

von Luxburg U. (2007). A tutorial on spectral clustering. *Statistics and Computing*, vol. 17, no. 4, pp. 395–416. DOI: 10.1007/s11222-007-9033-z.

Mikolov T. et al. (2013). Distributed Representations of Words and Phrases and their Compositionality. *Advances in Neural Information Processing Systems 26*. NIPS 2013. Lake Tahoe, NV, USA: Curran Associates, Inc., pp. 3111–3119. URL: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.

Newman M. E. J. (2004). Analysis of weighted networks. *Physical Review E*, vol. 70, no. 5, p. 056131. DOI: 10.1103/PhysRevE.70.056131.

Ng A., Jordan M., and Weiss Y. (2002). On Spectral Clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14*. NIPS 2002. Vancouver, BC, Canada: MIT Press, pp. 846–856. URL: https://proceedings.nips.cc/paper/2001/file/801272ee79cfde7fa5960571fee36b9b-Paper.pdf.

Panchenko A. et al. (2018). Building a Web-Scale Dependency-Parsed Corpus from Common Crawl. *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*. LREC 2018. Miyazaki, Japan: European Language Resources Association (ELRA), pp. 1816–1823. URL: https://aclanthology.org/L18-1286.

Pelevina M. et al. (2016). Making Sense of Word Embeddings. *Proceedings of the 1st Workshop on Representation Learning for NLP*. RepL4NLP. Berlin, Germany: Association for Computational Linguistics, pp. 174–183. DOI: 10.18653/v1/W16-1620.

References III

Peng Y., Choi B., and Xu J. (2021). Graph Learning for Combinatorial Optimization: A Survey of State-of-the-Art. *Data Science and Engineering*, vol. 6, no. 2, pp. 119–141. DOI: 10.1007/s41019-021-00155-3.

Shi J. and Malik J. (2000). Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905. DOI: 10.1109/34.868688.

Tauer G. et al. (2019). An incremental graph-partitioning algorithm for entity resolution. *Information Fusion*, vol. 46, pp. 171–183. DOI: 10.1016/j.inffus.2018.06.001.

Ustalov D. et al. (2019). Watset: Local-Global Graph Clustering with Applications in Sense and Frame Induction. *Computational Linguistics*, vol. 45, no. 3, pp. 423–479. DOI: 10.1162/COLI_a_00354.

Vlasblom J. and Wodak S. J. (2009). Markov clustering versus affinity propagation for the partitioning of protein interaction graphs. *BMC Bioinformatics*, vol. 10, no. 1, p. 99. DOI: 10.1186/1471-2105-10-99.

Supplementary Media I

Adamovich O. (September 3, 2015). Girls Whispering Best Friends. Pixabay. URL: `https://pixabay.com/images/id-914823/`. Licensed under Pixabay License.

Alexandrov O. (July 14, 2007). Illustration of a Vector space. Wikimedia Commons. URL: `https://commons.wikimedia.org/wiki/File:Vector_space_illust.svg`. Licensed under Public Domain.

bamenny (February 24, 2016). Robot Flower Technology. Pixabay. URL: `https://pixabay.com/images/id-1214536/`. Licensed under Pixabay License.

Buissinne S. (August 25, 2016). Dictionary Reference Book Learning. Pixabay. URL: `https://pixabay.com/images/id-1619740/`. Licensed under Pixabay License.

Finnsson I. (May 19, 2017). Books Covers Book Case. Pixabay. URL: `https://pixabay.com/images/id-2321934/`. Licensed under Pixabay License.

FreePhotosART (September 3, 2016). Cook Cooking School Pan. Pixabay. URL: `https://pixabay.com/images/id-1641959/`. Licensed under Pixabay License.

Kittner L. (October 26, 2015). Cook Cooking School Pan. Pixabay. URL: `https://pixabay.com/images/id-1002505/`. Licensed under Pixabay License.

Merrill B. (July 24, 2014). Pedestrians People Busy. Pixabay. URL: `https://pixabay.com/images/id-400811/`. Licensed under Pixabay License.

Rahman Rony M. (May 31, 2016). Mad Max Fury Car Monster. Pixabay. URL: `https://pixabay.com/images/id-1426796/`. Licensed under Pixabay License.

rawpixel (April 18, 2017). Calm Freedom Location. Pixabay. URL: `https://pixabay.com/images/id-2218409/`. Licensed under Pixabay License.