N-gram Graphs: A generic machine learning tool in the arsenal of NLP, Video Analysis and Adaptive Systems. (Part II)

George Giannakopoulos

1University of Trento, Italy
ggianna@disi.unitn.it

April 26, 2010
In the previous episode...

- N-gram Graphs and Operators
- Richer information
- Domain agnostic
- Generic applicability
- State-of-the-art performance in summary evaluation
- Promising for language-independent summarization
- Usable in classification, clustering, record linkage
This episode

- Representing behavior (using Optical Flow Proximity Graphs)
This episode

- Representing behavior (using Optical Flow Proximity Graphs)
- Combining N-gram Graphs with the Vector Space
This episode

- Representing behavior (using Optical Flow Proximity Graphs)
- Combining N-gram Graphs with the Vector Space
- User Modeling with N-gram Graphs
This episode

- Representing behavior (using Optical Flow Proximity Graphs)
- Combining N-gram Graphs with the Vector Space
- User Modeling with N-gram Graphs
- The JINSECT toolkit: An open source LGPL toolkit for N-gram Graphs
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
     - Whole Frame
     - Operators Revisited: Complexity
     - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Behavior Recognition

Examples

- Assistive Environment
Behavior Recognition

Examples

- Assistive Environment
- Super Market – Mall
Behavior Recognition

Examples

- Assistive Environment
- Super Market – Mall
- Parking Lot
Behavior Recognition

Examples

- Assistive Environment
- Super Market – Mall
- Parking Lot
- Vending Machines – ATMs
Behavior Recognition

Examples

- Assistive Environment
- Super Market – Mall
- Parking Lot
- Vending Machines – ATMs
- Traffic
Behavior Recognition

Examples

- Assistive Environment
- Super Market – Mall
- Parking Lot
- Vending Machines – ATMs
- Traffic
- Sports
Optical Flow

Image from http://api.ning.com/files/
DPSX6QXHN*m77We5ozsv1C1V7uw5qyicb90jUaDEda2vMbj*cnWX0m9T8YtCG61DU12ijCFR1n80fnvFHa0jWokU5EXwtKxE/sam200.jpg
Behavior Recognition and Video Indexing

The Method\textsuperscript{a}

\textsuperscript{a}In collaboration with Panagiota Antonakaki, NCSR Demokritos.
Behavior Recognition and Video Indexing

The Method

In collaboration with Panagiota Antonakaki, NCSR Demokritos.

- No a priori information required
Behavior Recognition and Video Indexing

The Method

In collaboration with Panagiota Antonakaki, NCSR Demokritos.

- No a priori information required
- No preprocessing steps required
Behavior Recognition and Video Indexing

The Method

- In collaboration with Panagiota Antonakaki, NCSR Demokritos.
- No a priori information required
- No preprocessing steps required
- Only optical flow for feature vector calculation
Representing Behavior — Variations

Proposed Methods

1. Whole frame representation using graphs (Optical Flow Proximity Graphs - OFPGs)
Representing Behavior — Variations

**Proposed Methods**

1. Whole frame representation using graphs (Optical Flow Proximity Graphs - OFPGs)
2. Segmentation and representation using hierarchy of graphs (Symbolic)
Outline

1 Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2 Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3 JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4 Closing
   - Summary and the Future
   - Appendix
Training Step

Video Analysis
Adaptive Systems
JINSECT: A Toolkit for All
Closing

The Optical Flow Proximity Graph
Whole Frame
Operators Revisited: Complexity
Hierarchy in Graphs

Whole Frame Representation (1)

George Giannakopoulos
N-Gram Graphs
Testing Step

Video Stream → Frame → Feature Vector

Optical Flow per frame

The Optical Flow Proximity Graph

Whole Frame Operators Revisited: Complexity Hierarchy in Graphs

Whole Frame Representation (2)
Extraction of feature vector.

Graph Representation from Vectors

\[ f = (\text{OF}_{\text{norm}}, \text{OF}_{\text{angle}}) \]  \hspace{1cm} (1)
where `getBinForValue` is a function that returns the name of the bin (quantization)
Parametrically Determined Window
What is Used?

Using N-gram Graph Operators

- Update operator
What is Used?

Using N-gram Graph Operators

- Update operator
- Comparison operator
What is Used?

Using N-gram Graph Operators

- Update operator
- Comparison operator
- Intersection operator
What is Used?

Using N-gram Graph Operators

- Update operator
- Comparison operator
- Intersection operator
- All-not-in or delta operator
Noise in Data

Reasons for removal

- Background noise due to camera
Noise in Data

Reasons for removal

- Background noise due to camera
- Classification lies in the differences between classes
Noise in Data

Reasons for removal

- Background noise due to camera
- Classification lies in the differences between classes

\[ g_{\text{Noise}} = \cap_{1 \leq j \leq N} \{ G_{c_j} \} \]

\[
\text{for each } G_{c_j} \text{ do} \\
| \quad \text{NoiselessClassGraph} = \text{NoiselessClassGraph}.\text{allNotIn}(g_{\text{Noise}}); \\
\text{end}
\]
Outline

1 Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
     - Hierarchy in Graphs
2 Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments
3 JINSECT: A Toolkit for All
   - Overview
   - Why use it?
4 Closing
   - Summary and the Future
   - Appendix
What is the Complexity of Trivially Implemented Graph Operators?

- Extraction from Source (of size $N$, $m$ dimensions):
  \[ O(D_{\text{win}}^m \times N) \]
- Similarities $|G_m| = \min(|G_1|, |G_2|)$, $|G^M| = \max(|G_1|, |G_2|)$
  - Size Similarity: $O(1)$
  - Containment and Value Similarity: $O(|G_m||G^M|)$
- Update, Merge: $O(|G_m||G^M| + |G_m|c)$
Improved Complexity of Graph Operators?

- Using hash for nodes (or edges)
- Quick search - slower insert
- Extraction from Text (of length \( N \)): \( D_{\text{win}} \times N \)
- Similarities \( |G_m| = \min(|G_1|, |G_2|), |G^M| = \max(|G_1|, |G_2|) \)
  - Size Similarity: \( O(1) \)
  - Containment and Value Similarity: \( O(|G_m| \log |G^M|) \)
  - Update, Merge: \( O(|G_m| \log |G^M| + |G_m|c) \)
N-gram Graph – Value Similarity

Example

```
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>1.0</td>
<td>8.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Result: \( \frac{1.0}{4} + \frac{4.0}{8} = \frac{1}{4} + \frac{1}{8} = 0.375 \)
Space Complexity and Related Considerations

- Vertices can be burdensome
- Edges can be burdensome
- Operators copying graphs
- Indexing increases memory requirement
Space Complexity and Related Considerations

- Vertices can be burdensome
- Edges can be burdensome
- Operators copying graphs
- Indexing increases memory requirement
- Serialization
Outline

1 Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2 Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3 JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4 Closing
   - Summary and the Future
   - Appendix

George Giannakopoulos

N-Gram Graphs
Segmented Frame Representation — Hierarchy (1)

Training Step

FIRST LEVEL

Video Stream → Frame → Segmented Frame → Feature Vector → Graph → Index of Graphs

Optical Flow per frame

OF_norm, OF_angle

George Giannakopoulos

N-Gram Graphs
Data: CurrentGraph, IndexOfGraphs
Result: informed IndexOfGraphs

dgName = null;
for each Graph in set of graphs of IndexOfGraphs do
    compute similarity between CurrentGraph and Graph;
    if similarity ≥ maxForMerging then
        dgName = name of the Graph;
    else if similarity ≥ minForMerging then
        dgName = name of the Graph;
        Graph = result of merging CurrentGraph and Graph;
    else
        CurrentGraph = result of removal of Graph from CurrentGraph
    end
end
if dgName = null then
    assign a new name to the CurrentGraph and add CurrentGraph and name in the index;
end
Segmented Frame Representation — Hierarchy (2)

Training Step

- Segmented Frame
- Graph
- Index of Graphs
- Second Level Graph
Segmented Frame Representation — Representing a Class

Training Step

Video Stream

Second Level Graph

SECOND LEVEL

Merging

Graph of Class

SVM Behavior

George Giannakopoulos

N-Gram Graphs
Segmented Frame Representation — Testing

Testing Step

FIRST LEVEL

Video Stream → Frame → Segmented Frame → Feature Vector → Graph → Index of Graphs

Optical Flow per frame

SECOND LEVEL

Index of Graphs → Graph

SVM Behavior1 → EXISTS/NOT_EXISTS

SVM Behavior2 → EXISTS/NOT_EXISTS

SVM BehaviorN → EXISTS/NOT_EXISTS

Behaviors

George Giannakopoulos

N-Gram Graphs
Frame as Symbols Example
Symbol Graph Example
Size of Index vs. Frames

![Graph showing the relationship between frame number and index size for different frame types: abrupt1, run, and walk.](image-url)
Experiments and Inter-class Similarity

Table 1: Before noise reduction inter-class similarity

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 2</th>
<th>Whole Frame Motion Representation Similarity</th>
<th>Symbolic Approach Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>walk</td>
<td>0.6644</td>
<td>0.4972</td>
</tr>
<tr>
<td>run</td>
<td>abrupt</td>
<td>0.6933</td>
<td>0.2001</td>
</tr>
<tr>
<td>walk</td>
<td>run</td>
<td>0.6644</td>
<td>0.4972</td>
</tr>
<tr>
<td>walk</td>
<td>abrupt</td>
<td>0.6738</td>
<td>0.3333</td>
</tr>
<tr>
<td>abrupt</td>
<td>run</td>
<td>0.6933</td>
<td>0.2001</td>
</tr>
<tr>
<td>abrupt</td>
<td>walk</td>
<td>0.6738</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

Table 2: After noise reduction inter-class similarity

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 2</th>
<th>Whole Frame Motion Representation Similarity</th>
<th>Symbolic Approach Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>walk</td>
<td>0.4039</td>
<td>0.4148</td>
</tr>
<tr>
<td>run</td>
<td>abrupt</td>
<td>0.4510</td>
<td>0.0522</td>
</tr>
<tr>
<td>walk</td>
<td>run</td>
<td>0.2891</td>
<td>0.4148</td>
</tr>
<tr>
<td>walk</td>
<td>abrupt</td>
<td>0.5494</td>
<td>0.2073</td>
</tr>
<tr>
<td>abrupt</td>
<td>run</td>
<td>0.2829</td>
<td>0.0522</td>
</tr>
<tr>
<td>abrupt</td>
<td>walk</td>
<td>0.3709</td>
<td>0.2073</td>
</tr>
</tbody>
</table>
### Noise Removal Effect

#### Table 1: Before noise reduction inter-class similarity

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 2</th>
<th>Whole Frame Motion Representation Similarity</th>
<th>Symbolic Approach Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>walk</td>
<td>0.6644</td>
<td>0.4972</td>
</tr>
<tr>
<td>run</td>
<td>abrupt</td>
<td>0.6933</td>
<td>0.2001</td>
</tr>
<tr>
<td>walk</td>
<td>run</td>
<td>0.6644</td>
<td>0.4972</td>
</tr>
<tr>
<td>walk</td>
<td>abrupt</td>
<td>0.6738</td>
<td>0.3333</td>
</tr>
<tr>
<td>abrupt</td>
<td>run</td>
<td>0.6933</td>
<td>0.2001</td>
</tr>
<tr>
<td>abrupt</td>
<td>walk</td>
<td>0.6738</td>
<td>0.3333</td>
</tr>
</tbody>
</table>

#### Table 2: After noise reduction inter-class similarity

<table>
<thead>
<tr>
<th>Category 1</th>
<th>Category 2</th>
<th>Whole Frame Motion Representation Similarity</th>
<th>Symbolic Approach Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>walk</td>
<td>0.4039</td>
<td>0.4148</td>
</tr>
<tr>
<td>run</td>
<td>abrupt</td>
<td>0.4510</td>
<td>0.0522</td>
</tr>
<tr>
<td>walk</td>
<td>run</td>
<td>0.2891</td>
<td>0.4148</td>
</tr>
<tr>
<td>walk</td>
<td>abrupt</td>
<td>0.5494</td>
<td>0.2073</td>
</tr>
<tr>
<td>abrupt</td>
<td>run</td>
<td>0.2829</td>
<td>0.0522</td>
</tr>
<tr>
<td>abrupt</td>
<td>walk</td>
<td>0.3709</td>
<td>0.2073</td>
</tr>
</tbody>
</table>
## Experiments (Semveillance dataset)

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Precision</th>
<th>Recall</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>run</td>
<td>0.9656</td>
<td>0.7178</td>
<td>0.8231</td>
</tr>
<tr>
<td>walk</td>
<td>0.6741</td>
<td>0.9287</td>
<td>0.7746</td>
</tr>
<tr>
<td>abrupt</td>
<td>0.9522</td>
<td>0.9298</td>
<td>0.9408</td>
</tr>
</tbody>
</table>
Experiments (PETS04 dataset [Fisher, 2004])

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Whole Frame Representation</th>
<th>With Frame Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Precision</td>
<td>Recall</td>
</tr>
<tr>
<td>browser</td>
<td>0.2093</td>
<td>0.3459</td>
</tr>
<tr>
<td>walker</td>
<td>0.9423</td>
<td>0.9491</td>
</tr>
<tr>
<td>fighters</td>
<td>0.1263</td>
<td>0.9461</td>
</tr>
<tr>
<td>meeters</td>
<td>0.2934</td>
<td>0.9810</td>
</tr>
</tbody>
</table>

Table 6: Experimental results for video indexing.

<table>
<thead>
<tr>
<th>Behavior</th>
<th>Whole Frame Representation</th>
<th>With Frame Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Specificity</td>
<td>Specificity</td>
</tr>
<tr>
<td>browser</td>
<td>0.3521</td>
<td>0.8444</td>
</tr>
<tr>
<td>walker</td>
<td>0.1829</td>
<td>0.8613</td>
</tr>
<tr>
<td>fighters</td>
<td>0.5257</td>
<td>0.8157</td>
</tr>
<tr>
<td>meeters</td>
<td>0.2605</td>
<td>0.8077</td>
</tr>
</tbody>
</table>
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Entity Name System (ENS) [Bouquet et al., 2008, Palpanas et al., 2008]

An **ENS**:
- Maps real world entities to *Unique* identifiers.
- Provides for the *reuse* of identifiers.
- Supports *disambiguation* to real world entities in the Web.
**Entity** in the ENS is a set of:

- Free-form attribute names
- Free-form attribute values

**Entity Example**

```
title : Dr
firstName : Themis
family_name : Palpanas
homepage : http://dit.unitn.it/~themis
affiliation : University of Trento
```
An **Adaptive Entity Subscription System (AESS)** provides for:

- management of subscription to specific entities.
- the update of subscribers over changes to entities.
- informing subscribers over changes they are *mostly interested in*.
- takes into account explicitly or implicitly declared user interests.
Change Examples - Type and Content

Type: Deletion (of entity)
Content: (N/A)
or
Type: Entity Update, Attribute Update
Content: title → Prof
or
Type: Entity Update, Attribute Insertion
Content: affiliation → University of Trento
Our AESS

- expects user feedback for interest indication.
- expresses interest as a real value.
- defines predefined values for interest levels.

We need to
- create an architecture for the system.
Our AESS

- expects user feedback for interest indication.
- expresses interest as a real value.
- defines predefined values for interest levels.

We need to

- create an architecture for the system.
- represent efficiently the type and content (i.e. free form strings) info of a change.
Our AESS

- expects user feedback for interest indication.
- expresses interest as a real value.
- defines predefined values for interest levels.

We need to

- create an architecture for the system.
- represent efficiently the *type* and *content (i.e. free form strings)* info of a change.
- create a user model, from user feedback, that can use this representation.
Our AESS

- expects user feedback for interest indication.
- expresses interest as a real value.
- defines predefined values for interest levels.

We need to

- create an architecture for the system.
- represent efficiently the *type* and *content (i.e. free form strings)* info of a change.
- create a user model, from user feedback, that can use this representation.
- take into account both simple and complex scenarios of preference.
Architecture

- Change Queue
  - Change Info
    - Consumers
  - User Feedback
- ENS DB
- Entity Change Trigger
Architecture

- Change Queue
- User Profile DB
Architecture

- Change Queue
- User Profile DB
- Adaptive Information Control
Architecture

- Change Queue
- User Profile DB
- Adaptive Information Control
- Subscription Information Broker
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
The **type of a change**:  
- deletion, splitting, merging or update  
- updates can involve: attribute deletion, attribute insertion or attribute update  
- given graded indication of normality of the change, e.g. 0 (abnormal) to 1 (normal)

**Feature Space**

A dimension indicative of each type/subtype of change.

But what about *Content*?
Representing Changes — Content

The **content of a change:**

- Instances of attribute names
- Instances of attribute values

**Problems and requirements**

- Free form strings
- Other types (numeric, date, etc.)
- Fuzzy string matching
- Updatable model — if possible
- Graded similarity from comparison of instance to model

We use **Character N-gram Graphs**
A character n-gram graph is a string model based on the coexistence of character n-grams in a string.

first_name: Basil

Graph Size: 39 bidirectional edges
first_name:Basil
first_name:George

Graph Size: 42 bidirectional edges

i.e., not bad scaling for normal user requirements.
Text Size to Graph Size — Actual vs Random

Actual Text

Random Text
Mapping Content to the Feature Space

Given

*a set of labeled changes and a new change.*

We want

*dimensions* indicative of content similarity.

**N-gram Graph Normalized Value Similarity (NVS)**

- Create a graph representing labeled instances for each level.
- We have one similarity-based feature for each interest level.
Outline

1 Evolution of N-gram Graphs to Video Analysis
   • The Optical Flow Proximity Graph
   • Whole Frame
   • Operators Revisited: Complexity
   • Hierarchy in Graphs

2 Modeling User Preferences
   • Overview of Senses
   • Representation
   • Overview of Solution
   • Data and Experiments

3 JINSECT: A Toolkit for All
   • Overview
   • Why use it?

4 Closing
   • Summary and the Future
   • Appendix
To map a change instance to the feature space:

- Apply values to *Type* dimensions.
- Calculate *Content* graph similarities for every interest level.
- Add dimensions for *Content* graph similarity.
Update Model with New Data

To update the user model with a new instance:

- Merge *Content*-based graph into corresponding interest level graph.
- Initialize new vector.
- Calculate an $\epsilon$-SVR [Chang and Lin, 2001, Vapnik, 1998] regression model to estimate interest.

How does this model perform?
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Experimental Setting

- Synthetic data for data changes
- 10-fold validation
- 1000 changes per fold
- Each iteration is mapped to a set of 10 changes

We judge

- if learning occurs and its rate.
- if the use of content (graphs) is useful.
### Determining Learning Ability

**Table:** Correlation between Emission-Iteration Number and Regression Mean Absolute Error per Subscriber Profile and Method

<table>
<thead>
<tr>
<th>Subscriber</th>
<th>Graphs</th>
<th>Correlation (p-value)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type-based</td>
<td>✓</td>
<td>-0.3398559 (&lt; 10(^{-2}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.2993715 (&lt; 10(^{-2}))</td>
</tr>
<tr>
<td>Attribute name-based</td>
<td>✓</td>
<td>-0.3734062 (&lt; 10(^{-2}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.03564642 (0.2601)</td>
</tr>
<tr>
<td>Attribute name-value pair-based</td>
<td>✓</td>
<td>-0.0858178 (&lt; 10(^{-2}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.02968072 (0.3484)</td>
</tr>
<tr>
<td>Complex</td>
<td>✓</td>
<td>-0.5989662 (&lt; 10(^{-2}))</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-0.03393356 (0.2837)</td>
</tr>
</tbody>
</table>
Rate of Acceptable Errors (RAE): Percentage of errors in estimation that cannot cause ranking error.


**Figure:** Type Based

**Figure:** Attribute Name-based

Determining Speed and Stability of Learning (1)
Rate of Acceptable Errors (RAE): Percentage of errors in estimation that cannot cause ranking error.


Figure: Attribute Name-Value-based

Figure: Complex
Importance of Representation

Rate of Acceptable Errors (RAE): Percentage of errors in estimation that cannot cause ranking error.

**Figure:** Name, Value in One Graph

**Figure:** Name, Value in Separate Graphs
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Application suite

- AutoSummENG (plus new version)
- MUDOS-NG
- Document Classifier
- Spam filter
- Grammaticality Estimator
- Entropy-based Chunk Splitter
Library

- Character and Word N-gram Graphs
- N-Gram Distribution Graphs
- Operators
- **Serializability**
- Distributed Processing Examples (JADE)
- Multi-threading
- Utilities (file to string, Distribution class, etc.)
- Interoperability (R, thesauri, etc.)
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Open Source

- LGPL
- Extendable
- Reusable
- Lots of examples
- Non-trivial implementations
Easy to Apply

- Find what the vertices should be
- Define the neighborhood relation
- Use them
Outline

1. Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2. Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3. JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4. Closing
   - Summary and the Future
   - Appendix
Almost there...

### Flashback
- Optical Flow Proximity Graphs
- Proximity Graphs in a Hierarchy
- Combining Graphs with Vector Space
- JINSECT Toolkit and Library
Into the future...

- Indexing graphs
- Hierarchy and granularity criteria
- Expressiveness of proximity graph
- Recognition of n-gram graphs
Into the future...

- Indexing graphs
- Hierarchy and granularity criteria
- Expressiveness of proximity graph
- Recognition of n-gram graphs
- ...and whatever *you* plan to make out of them.
Thank you
George Giannakopoulos (ggianna@disi.unitn.it)

Please provide your thoughts on the feedback form\(^1\).

\(^1\)See http://tinyurl.com/2fna572
Outline

1 Evolution of N-gram Graphs to Video Analysis
   - The Optical Flow Proximity Graph
   - Whole Frame
   - Operators Revisited: Complexity
   - Hierarchy in Graphs

2 Modeling User Preferences
   - Overview of Senses
   - Representation
   - Overview of Solution
   - Data and Experiments

3 JINSECT: A Toolkit for All
   - Overview
   - Why use it?

4 Closing
   - Summary and the Future
   - Appendix
## Change Data Generation: User Simulation

<table>
<thead>
<tr>
<th>User type (Prob.)</th>
<th>Change type</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Benevolent (0.95)</td>
<td>Attribute change (normal)</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Attribute insertion</td>
<td>0.30</td>
</tr>
<tr>
<td></td>
<td>Attribute deletion</td>
<td>0.10</td>
</tr>
<tr>
<td>Sys.admin.(0.03)</td>
<td>Entity merge</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>Entity split</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td>Entity deletion</td>
<td>0.10</td>
</tr>
<tr>
<td>Malevolent (0.02)</td>
<td>Attribute change (abnormal)</td>
<td>0.70</td>
</tr>
<tr>
<td></td>
<td>Attribute deletion</td>
<td>0.30</td>
</tr>
<tr>
<td>Subscriber</td>
<td>Importance</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Type-based</td>
<td>Critical</td>
<td>Attribute deletion.</td>
</tr>
<tr>
<td></td>
<td>Interesting</td>
<td>Entity deletion.</td>
</tr>
<tr>
<td>Attribute name-based</td>
<td>Critical</td>
<td>Any change concerning an attribute that contains the string “name”.</td>
</tr>
<tr>
<td></td>
<td>Interesting</td>
<td>(None)</td>
</tr>
<tr>
<td>Attribute name-value pair-based</td>
<td>Critical</td>
<td>Attribute change or insertion on “isDeceased” attribute, with a new value of “true”.</td>
</tr>
<tr>
<td></td>
<td>Interesting</td>
<td>Attribute change or insertion on “isDeceased” attribute, with a new value of “false”.</td>
</tr>
<tr>
<td>Complex</td>
<td>Critical</td>
<td>Default attribute (some attributes in the ENS are considered default — e.g., the name of a person entity — while all the others non-default) update or insertion with an abnormal value.</td>
</tr>
<tr>
<td></td>
<td>Interesting</td>
<td>Default attribute deletion or normal update.</td>
</tr>
</tbody>
</table>

