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

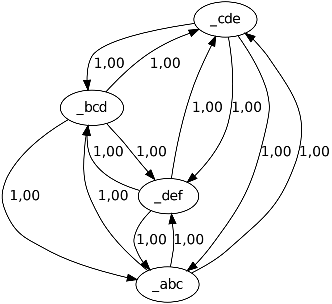
George Giannakopoulos[1]

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

April 27, 2010

Outline

An N-gram Graph



What does an n-gram graph do?

- Indicates neighborhood
- **Edges** are important
- Edge weights can have different semantics

Intuition — Perception

- People can read even when words are spelled *wnorg*

Intuition — Perception

- People can read even when words are spelled *wnorg*
- But order *does* play some role: *not it does?*

Intuition — Perception

- People can read even when words are spelled *wnorg*
- But order *does* play some role: *not it does?*
- The same stands for images...

Intuition — Perception

- People can read even when words are spelled *wnorg*
- But order *does* play some role: *not it does?*
- The same stands for images...
- ... and audio.

Intuition — Assumptions

- *Neighborhood* (or *Proximity*) can indicate relation, or causality

Intuition — Assumptions

- *Neighborhood* (or *Proximity*) can indicate relation, or causality
- An object can be analyzed into its constituent items

Intuition — Assumptions

- *Neighborhood* (or *Proximity*) can indicate relation, or causality
- An object can be analyzed into its constituent items
- You can compute *similarity* or *distance* for the domain of these items.

Intuition — Assumptions

- *Neighborhood* (or *Proximity*) can indicate relation, or causality
- An object can be analyzed into its constituent items
- You can compute *similarity* or *distance* for the domain of these items.
- Size of the neighborhood: level of description.

Extraction Process (NLP example)

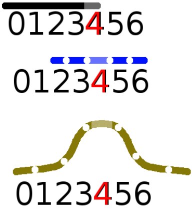
- Extract n-grams of ranks $[L_{\text{min}}, L_{\text{MAX}}]$. One graph per rank.
- Determine neighborhood (window size D_{win}).
- Assign weights to edges.

Example

String:	*abcde*
Character N-grams (Rank 3):	*abc, bcd, cde*
Edges (Window Size 1):	*abc-bcd, bcd-cde*
Weights (Occurrences):	*abc-bcd (1.0) , bcd-cde (1.0)*

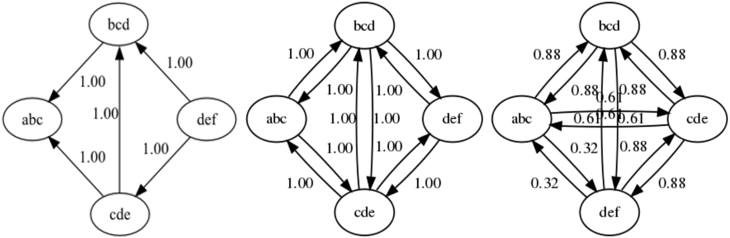
Window-based Extraction of Neighborhood — Examples

Figure: N-gram Window Types (top to bottom): non-symmetric, symmetric and gauss-normalized symmetric. Each number represents either a word or a character n-gram



N-gram Graph — Representation Examples

Figure: Graphs Representing the String *abcdef* (from left to right): non-symmetric, symmetric and gauss-normalized symmetric. N-Grams of Rank 3. D_{win} value 2.



What is an N-gram Graph?

- Restrictions upon relative positioning
- Correspond to all texts that comply with the restrictions
- Smaller distance, less generalization / fuzziness

What is Common with Existing Approaches

- Instance and class: common representation
- Updatable model

What is New with the N-gram Graph

- Co-occurrence information inherent
- Arbitrary fuzziness based on a parameter
- Generic applicability (domain agnostic) due to operators

...and more.

Frequently Asked Questions

Why not bag-of-words? Much more information

What about preprocessing (stemming, lemmatization, etc.)? Not needed

Are N-gram Graphs probabilistic? Not necessarily

Are N-gram Graphs automata for recognition? No, vertices are not states

Are they grammar models? Can be

I see too many parameters... Optimizable a priori

Outline

N-Gram Graph Generic Operators (1)

- Merging or Union $\cup$
- Intersection $\cap$
- Delta Operator (*All-Not-In* operator) $\triangle$
- Inverse Intersection Operator $\bigtriangledown$

N-Gram Graph Generic Operators (2)

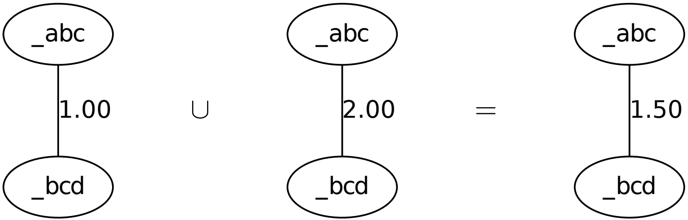
- Similarity function sim
- Update U (Merging is a special case of update)
- Degradation $\searrow$

Representing Sets of Graphs

A representative graph for a set

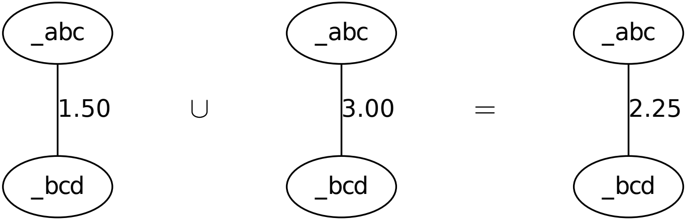
- is similar to functionality to the centroid of vectors
- **cannot** be represented using the **merging** operator (unless trivial)
- **can** be represented using the **update** operator
- for non-common edges the effect is **non-linear**

Merge vs. Update (1)



$$\frac{1+2}{2} = 1.5$$

Merge vs. Update (2)



$$\frac{1.5+3}{2} = 2.25$$

But what if we want $\frac{1+2+3}{3} = 2$?

Merge vs. Update (3)

$$\text{updatedValue} = \text{oldValue} + l \times (\text{newValue} - \text{oldValue}) \qquad (1)$$

where $0 \leq l \leq 1$ is the learning factor

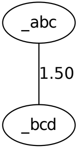
Representative (or *Centroid*) Graph

Use update operator with learning factor: $\frac{1}{\text{instanceCount}}$, where *instanceCount* is the number of instances that will be described by the graph *after* the update.

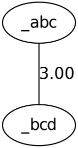
Merge vs. Update (4)

$$l = \frac{1}{3}$$

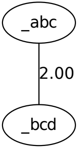
update(



,



$, l) =$



$$1.5 + \frac{1}{3} \times (3 - 1.5) = \frac{3}{2} + \frac{1}{3} \times \frac{3}{2} = \frac{4}{2} = 2$$

N-gram Graph – Similarity (1)

- Size Similarity: Number of Edges
- Co-occurrence Similarity: **Existence** of Edges
- Value Similarity: **Existence** and **Weight** of Edges
- Derived Measures: Normalized Value Similarity
- Similarity measures are symmetric (with some exceptions)
- Overall similarity: Weighted Normalized Sum over All N-Gram Ranks

N-gram Graph – Similarity (2)

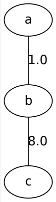
- T_i maps a set of graphs $\mathbb{G}^r$
- Size Similarity of G_1, G_2: $\mathrm{SS} = \frac{\min(|G_1|,|G_2|)}{\max(|G_1|,|G_2|)}$
- Containment Similarity: Each common edge adds $\frac{1}{\min(|G_1|,|G_2|)}$ to a sum.
- Value Similarity: Using weights, every common edge adds

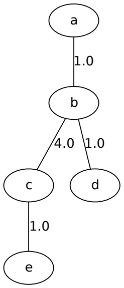
$$\frac{\frac{\min(w_e^i, w_e^j)}{\max(w_e^i, w_e^j)}}{\mathrm{SS}}$$

- Normalized Value Similarity, SS factored out: $\mathrm{NVS} = \frac{VS}{SS}$
- Overall Similarity for $n \in [L_{\min}, L_{\mathrm{MAX}}]$: Weighted sum of rank similarity.

N-gram Graph – Size Similarity

Example



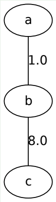


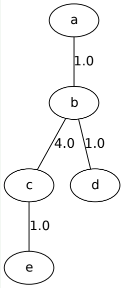
$|G_1| = 2$ $|G_2| = 4$

Result: $\frac{2}{4} = 0.5$

N-gram Graph – Containment Similarity

Example

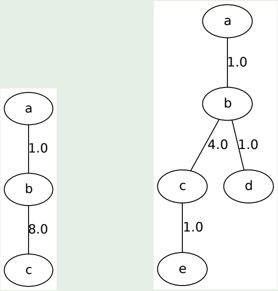




Result: $\frac{1}{2} + \frac{1}{2} = 1.0$

N-gram Graph – Value Similarity

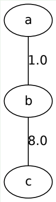
Example

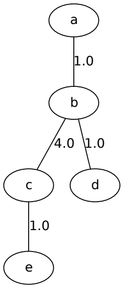


Result: $\frac{\frac{1.0}{1.0}}{4} + \frac{\frac{4.0}{8.0}}{4} = \frac{1}{4} + \frac{1}{8} = 0.375$

N-gram Graph – Normalized Value Similarity

Example





Result: $\frac{VS}{SS} = \frac{0.375}{0.5} = 0.75$

Outline

1. The N-gram Graph — Overview and Framework
   - Introducing the N-gram Graph
   - N-Gram Graph Generic Operators
   - N-Gram Graphs: Defining Noise
   - Applications
2. NLP Using N-gram Graphs
   - Introduction
   - Summary Evaluation
   - Optimizing N-gram Graph Parameters
   - Text Summarization
   - Other NLP Applications
3. Closing
   - Summary and Sneak Peek
   - Appendix

Noise – Definition

The definition of noise is task based, *e.g.*:

- Classification – Common inter-class graph.
- Text representation – 'Stopword' effect edges.

Can the noise be easily detected and removed? **Yes** through simple graph operators.

Noise – Questions

- *How can one determine the maximum common subgraph between classes?* – Intersection operator.
- *Is this (sub)graph unique?* – No, it is not.
- *Noisy subgraph approximation?* – Yes. The noisy subgraph can be easily approximated in very few steps.
- *Is the removal of the noisy (sub)graph effective?* – Yes.

Effect of graph noise removal (1)

The task

- Sets of texts, each from a different topic
- Create representative (centroid) graph per class
- Using training instances assign each doc to maximally similar class

NOTE: The maxarg operator is trivial for classification

Effect of graph noise removal (2)

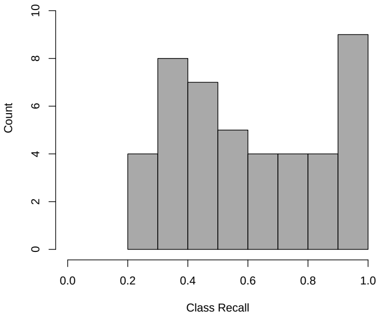


Figure: Class recall histogram for the classification task *including* noise

Effect of graph noise removal (3)

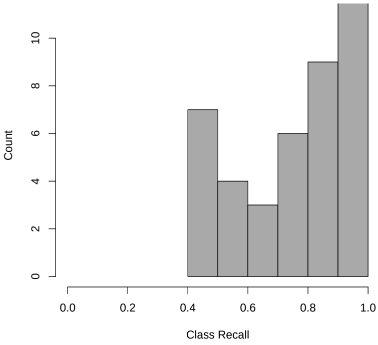


Figure: Class recall histogram for the classification task *without* noise

Outline

1. The N-gram Graph — Overview and Framework
   - Introducing the N-gram Graph
   - N-Gram Graph Generic Operators
   - N-Gram Graphs: Defining Noise
   - Applications
2. NLP Using N-gram Graphs
   - Introduction
   - Summary Evaluation
   - Optimizing N-gram Graph Parameters
   - Text Summarization
   - Other NLP Applications
3. Closing
   - Summary and Sneak Peek
   - Appendix

Example Domains

- *NLP:* Texts, words, sentences, whole texts

Example Domains

- *NLP:* Texts, words, sentences, whole texts
- *Pattern Matching:* Time series, short sequences, long sequences

Example Domains

- *NLP:* Texts, words, sentences, whole texts
- *Pattern Matching:* Time series, short sequences, long sequences
- *Natural Science:* Events, simple events, complex events

Example Domains

- *NLP:* Texts, words, sentences, whole texts
- *Pattern Matching:* Time series, short sequences, long sequences
- *Natural Science:* Events, simple events, complex events
- *Social Science:* Relations, groups, community

Text Classification – Spam filtering

CEAS 2008[1] Classify 140000 e-mails as spam or ham (on-line feedback).

		Gold Standard	
		Ham	Spam
Filter	Ham	24900	1777
Result	Spam	2229	108799
	Total	27129	110576

Percentages:

Ham% 8.22

Spam% 1.61

Using the trivial maxarg operator for the decision. *Extremely few* instances needed (tens).

[1] See `http://www.ceas.cc/2008/challenge/` for more on the challenge.

Record Linkage and Text Stemmatology (1)

Lineage of texts or record descriptions

- Compare all to all texts/records
- Determine threshold of similarity and *cluster*
- Create latent parents, through merging

Record Linkage and Text Stemmatology (2)

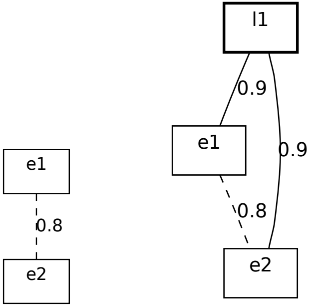


Figure: A case of latent node addition

Outline

Intuition — Language Analysis

- One thing that is common in all languages: context

Intuition — Language Analysis

- One thing that is common in all languages: context
- All characters are important in a text

Intuition — Language Analysis

- One thing that is common in all languages: context
- All characters are important in a text
- Even delimiters play a role in meaning

Intuition — Language Analysis

- One thing that is common in all languages: context
- All characters are important in a text
- Even delimiters play a role in meaning
- A word is a neighborhood of characters

Outline

The problem

Given *a set of model summaries*, determine the *quality* of a given *peer* summary text.

The problem

Given *a set of model summaries*, determine the *quality* of a given *peer* summary text.

Solution

- Represent all texts as n-gram graphs.
- Version 1: Compare between models and peer text and extract the average similarity.
- Version 2: **OR** merge models and compare peer text to merged model.

Indicative of responsiveness.

Overview of AutoSummENG

- Statistical *i.e.* **Language-Neutral**
- Word N-gram or **Character** N-Gram (Q-Gram) Based
- Graph Based on Neighborhood *i.e.* Includes Uncertainty / Fuzziness
- **No Preprocessing**

AutoSummENG method [Giannakopoulos et al., 2008]: State-of-the-art DUC 2005-2007, TAC 2008-2010

AutoSummENG – Evaluation Over DUC & TAC

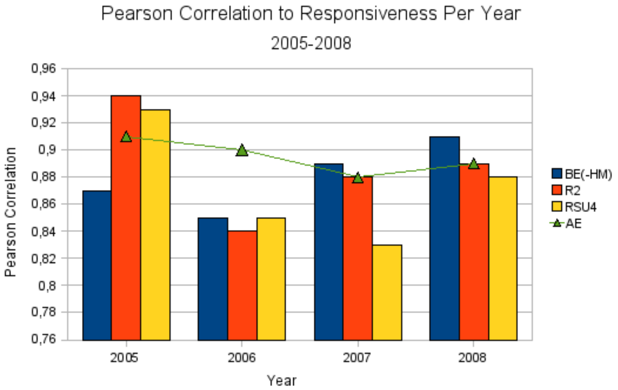


Figure: Pearson Correlation: Measures to (Content) Responsiveness for peers only

Outline

Optimizing Parameters (1)

- Minimum n-gram length, indicated as L_{min}.
- Maximum n-gram length, indicated as L_{MAX}.
- Neighborhood Window Size, indicated as D_{win}.

Optimizing Parameters (2)

Signal-to-Noise Optimization [Giannakopoulos et al., 2008]

- Symbols (Signal): contain letters neighboring more often than random characters
- Non-symbols (Noise): the rest

Signal-to-Noise: Elaboration (1)

We count, given a corpus T_0:

- times X appears in T_0, represented by N_X.
- how many times the string Xy appears in T_0, represented by N_{Xy}.
- the total number of n-grams of a given size n within T_0, represented by $|T_{0,n}|$.

Signal-to-Noise: Elaboration (2)

- $P(y|X)$ of a given suffix y, given the prefix X is

$$P(y|X) = P(X) * P(y, X)$$

where $P(y, X) = \frac{N_{Xy}}{|T_{0,n}|}$ and $P(X) = \frac{N_X}{|T_{0,|X|}|}$

- Random sequence probability is

$$P(y_r|X) = P(y_r)$$

Signal-to-Noise: Elaboration (3)

Signal-to-Noise

$$SN(L_{\mathrm{min}}, L_{\mathrm{MAX}}) = 10 \times \log_{10}\left(\frac{S(L_{\mathrm{min}}, L_{\mathrm{MAX}})}{N(L_{\mathrm{min}}, L_{\mathrm{MAX}})}\right)$$

$$N(L_{\mathrm{min}}, L_{\mathrm{MAX}}) = \sum_{i=L_{\mathrm{min}}}^{L_{\mathrm{MAX}}} |\text{Non-Symbols}_i|$$

Signal is more complex: Importance of symbols is related to their length. *Weighted symbols* are calculated and redistributed over ranks (Appendix — Slide 5).

Extracting Symbols

Input: text T_0^L
Output: symbol set S
// t denotes the current iteration
// $T[i]$ denotes the i-th character of T
// ϵ is the empty string
// $P(y_r)$ is the probability of a random suffix y_r
// The plus sign (+) indicates concatenation where character series are concerned.

$S = \emptyset$;
$s_t = T_0^L[1]$;
for *all i in [2,length(T_0^L)]* **do**
    $y = T_0^L[i]$;
    $c_t = s_t + y$;
    if $P(y|s_t) > P(y_r)$ **then**
        $s_t = c_t$;
    end
    else
        $S = S + s_t$;
        $s_t = y$;
    end
end
// Add last symbol
$S = S + s_t$;

Extracting Symbols — Example

Text: *Trying to understand...*

1st Step

$$s_t =' T'$$

$$y =' r'$$

$$P(y_r) = \frac{1}{64}, P(y|s_t) = \frac{1}{60}$$

Extracting Symbols — Example

Text: *Trying to understand...*

2nd Step

$$s_t =' Tr'$$

$$y =' y'$$

$$P(y_r) = \frac{1}{64}, P(y|s_t) = \frac{1}{40}$$

Extracting Symbols — Example

Text: *Trying to understand...*

3rd Step

$$s_t =' Try'$$

$$y =' \quad '$$

$$P(y_r) = \frac{1}{64}, P(y|s_t) = \frac{1}{80}$$

Extracting Symbols — Example

Text: *Trying to understand...*

New Symbol

$$s_t =' \ '$$

$$y =' t'$$

and so on...

Optimizing Parameters (4)

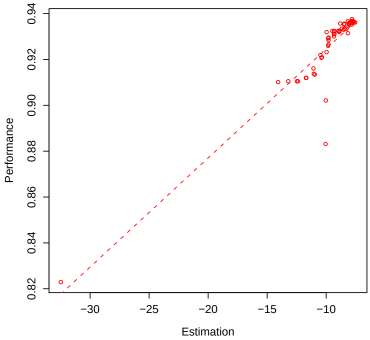


Figure: Correlation between Estimation (SN) and Performance

Outline

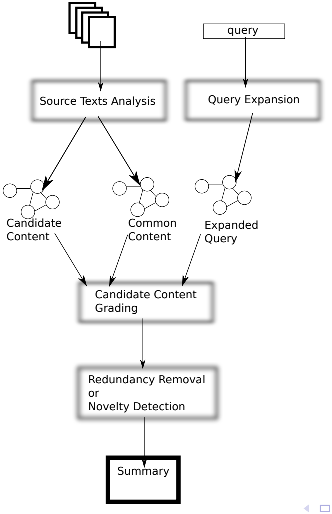
Summarizing Multiple Documents

- Given a set of texts referring to a subject
- Output summary

Summarizing Multiple Documents

- Given a set of texts referring to a subject
- Output summary
- Capture salient information
- Avoid redundancy

The MUDOS-NGSystem



Per Subtask Strategy

- Common Content: Intersection
- Expanded Query: Query expanded with synonyms
- Chunk Grading: Similarity
- Redundancy Checking: Similarity of Chunk vs.
  - other chunks
  - iteration summary text

Novelty Detection Algorithm

1. Extract the n-gram graph representation of the summary so far, indicated as G_{sum}.
2. Keep the part of the summary representation that does not contain the common content of the corresponding document set $\mathbb{U}$, $G'_{\text{sum}} = G_{\text{sum}} \triangle C_{\mathbb{U}}$.
3. For every candidate sentence in $\mathbb{L}$ that has not been already used
   1. extract its n-gram graph representation, G_{cs}.
   2. keep only $G'_{cs} = G_{cs} \triangle C_{\mathbb{U}}$, because we expect to judge redundancy for the part of the n-gram graph that is not contained in the common content $C_{\mathbb{U}}$.
   3. assign the similarity between G'_{cs}, G'_{sum} as the sentence redundancy score.
4. For all candidate sentences in $\mathbb{L}$
   1. Set the score of the sentence to be its rank based on the similarity to $C_{\mathbb{U}}$ minus the rank based on the redundancy score.
5. Select the sentence with the highest score as the best option and add it to the summary.
6. Repeat the process until the word limit has been reached or no other sentences remain.

MUDOS-NG Performance (1)

System (DUC 2006 SysID)	*AutoSummENG Score*
Baseline (1)	0.1437
Top Peer (23)	0.2050
Last Peer (11)	0.1260
Peer Average (All Peers)	0.1842 (Std. Dev. 0.0170)
Proposed System (-)	**0.1783**

Table: AutoSummENG performance data for DUC 2006. NOTE: The top and last peers are based on the AutoSummENG measure performance of the systems.

MUDOS-NG Performance (2)

System (TAC 2008 SysID)	*AutoSummENG Score*
Top Peer (43)	0.1991
Last Peer (18)	0.1029
Peer Average (All Peers)	0.1648 (Std. Dev. 0.0216)
Proposed System (-)	**0.1303**

Table: AutoSummENG performance data for TAC 2008. NOTE: The top and last peers are based on the AutoSummENG measure performance of the systems.

Outline

Sentiment Analysis [Rentoumi et al., 2009]

- Classification task
- Positive vs. Negative thesaurus sense descriptions
- Polarity assignment based on similarity of word sense to classes

Semantic Annotation [Giannakopoulos, 2009]

- Symbolic Graph
  - Indicates substring relations as a tree
  - Mapping strings to sense descriptions or synonyms (from thesaurus)
- A string is assigned the union of meanings of its substrings

Semantic Similarity (1)

$$\text{rel}_{\text{Meaning}}(t_1, t_2) = \frac{\sum_{G_{1i}, G_{2j}} \text{sim}(G_{1i}, G_{2j})}{|D_1| \times |D_2|}$$

Similarity: the average similarity between all pairs of senses.

Semantic Similarity (2)

t_1	t_2	$\text{rel}_{\text{Meaning}}$
run	jump	0.0017
smart	stupid	0.0020
run	walk	0.0020
smart	pretty	0.0036
smart	clever	0.0000
hollow	empty	0.1576
run	operate	0.2162
hollow	holler	0.3105

Outline

Almost there...

- N-gram Graphs and Operators
- Richer information
- Domain agnostic
- Generic applicability

Almost there...

- 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
- ...and others

Sneak Peek in Second Part

- Representing behavior with N-gram Graphs

Sneak Peek in Second Part

- Representing behavior with N-gram Graphs
- Combining N-gram Graphs with the Vector Space

Sneak Peek in Second Part

- Representing behavior with N-gram Graphs
- Combining N-gram Graphs with the Vector Space
- User modeling with N-gram Graphs

Sneak Peek in Second Part

- Representing behavior with N-gram 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

Looking forward to seeing you in the second part

Thank you

Outline

AutoSummENG – Evaluation TAC 2008

AE to...	*Spearman*	*Kendall*	*Pearson*
Resp.	0.8953 (< 0.01)	0.7208 (< 0.01)	0.8945 (< 0.01)
Ling.	0.5390 (< 0.01)	0.3819 (< 0.01)	0.5307 (< 0.01)

Table: Correlation of the *system* AutoSummENG score to human judgment for peers only (p-value in parentheses)

AE to ...	*Spearman*	*Kendall*	*Pearson*
Resp.	0.3788 (< 0.01)	0.2896 (< 0.01)	0.3762 (< 0.01)
Ling.	0.1982 (< 0.01)	0.1492 (< 0.01)	0.1933 (< 0.01)

Table: Correlation: *Summary* AutoSummENG to human judgment for peers only

MUDOS-NG Variations' Performance

System ID	CS	SS	RR	ND	QE	NE	Score
1		✓		✓		✓	0.1202
2		✓	✓			✓	**0.1303**
3	✓		✓		✓		0.1218
4		✓		✓	✓		0.1198
5		✓	✓		✓		0.1299
6	✓					✓	0.1255

Table: AutoSummENG summarization performance for different settings concerning scoring, redundancy and query expansion. **Legend** CS: Chunk Scoring, SS: Sentence Scoring, RR: Redundancy Removal, ND: Novelty Detection, QE: Query Expansion, NE: No Expansion. Best performance in **bold**.

Symbols and Non-symbols

'permanent', 'permit', 'permits', 'persist', 'person', 'personal', 'personal computers', 'personnel,' 'persons', 'persuade', 'pesticide', 'pesticides.',
'permi', 'permitt', 'pers', 'pers and', 'person kn', 'person or', 'perti', 'perty', 'pes', 'pes o'

Figure: Sample extracted symbols

'permit </HEADLINE>', 'permit program', 'permit approved'

Figure: Sample non-symbols

Signal Calculation

The number of weighted symbols for each n-gram rank r is calculated in two steps, within the given range $[L_{\min}, L_{\text{MAX}}]$:

1. Calculate the weight w_r of symbols for the specified rank r and sum over all weighted symbols to find the total, *weighted symbol sum* W_r for rank r. The weight w_s is defined to be the inverse of the probability of producing a symbol of rank r given a symbol of rank $r-1$, as longer symbols are less probable to appear as a result of a *random sampling* of characters. This means that we consider more important sequences that are less likely to have been randomly produced. Thus:

$$P(s_r|s_{r-1}) = \begin{cases} \frac{1}{|\text{Symbols}_r|+|\text{Non-Symbols}_r|} & \text{if } r = 1. \\ \frac{1}{|\text{Symbols}_{r-1}|+|\text{Non-Symbols}_{r-1}|} \times \frac{1}{|\text{Symbols}_r|+|\text{Non-Symbols}_r|} & \text{else.} \end{cases}$$

So $w_r = 1/P(s_r|s_{r-1})$ (2)

where $|\text{Symbols}_r|$ is the number of symbols in rank r.

2. Normalize W_r so that the sum of W_r over $r \in [L_{\min}, L_{\text{MAX}}]$ is equal to the original number of symbols in the texts. The normalized, weighted symbols W_r^0 for rank r are calculated by:

$$W_r^0 = W_r \times \frac{|\text{Symbols}_r|}{\sum_{i=L_{\min}}^{L_{\text{MAX}}} |\text{Symbols}_i|} \quad (3)$$

We indicate once more that the W_r^0 measure actually represents the *importance of symbols* per rank r for the symbols of the texts, instead of the *number of symbols* per rank that is indicated by $|Symbols_r|$.

Giannakopoulos, G. (2009).
Automatic Summarization from Multiple Documents.
PhD thesis, Department of Information and Communication Systems Engineering, University of the Aegean, Samos, Greece, http://www.iit.demokritos.gr/~ggianna/thesis.pdf.

Giannakopoulos, G., Karkaletsis, V., Vouros, G., and Stamatopoulos, P. (2008).
Summarization system evaluation revisited: N-gram graphs.
ACM Trans. Speech Lang. Process., 5(3):1–39.

Rentoumi, V., Giannakopoulos, G., Karkaletsis, V., and Vouros, G. (2009).
Sentiment analysis of figurative language using a word sense disambiguation approach.
Borovets, Bulgaria.