ΕΠΙΛΕΓΜΕΝΕΣ

ΠΤΥΧΙΑΚΕΣ & ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Τόμος 14

ΑΘΗΝΑ
2017

ΕΠΙΛΕΓΜΕΝΕΣ ΠΤΥΧΙΑΚΕΣ & ΔΙΠΛΩΜΑΤΙΚΕΣ ΕΡΓΑΣΙΕΣ

Εκδίδεται μία φορά το χρόνο από το:

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών,
Πανεπιστημιούπολη, 15784 Αθήνα

Επιμέλεια έκδοσης:
Επιτροπή Ερευνητικών και Αναπτυξιακών Δραστηριοτήτων
Θ. Θεοχάρης (υπεύθυνος έκδοσης), Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Η. Μανωλάκος, Καθηγητής, Τμήμα Πληροφορικής και Τηλεπικοινωνιών

Γραφιστική επιμέλεια - Επιμέλεια κειμένων:
Ε. Φλωριάς, Τμήμα Πληροφορικής και Τηλεπικοινωνιών

ISSN 1792-8826

Περιεχόμενα

Πρόλογος

Ο τόμος αυτός περιλαμβάνει περιλήψεις επιλεγμένων διπλωματικών και πτυχιακών εργασιών που εκπονήθηκαν στο Τμήμα Πληροφορικής και Τηλεπικοινωνιών του Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών κατά το διάστημα **01/01/2016 - 31/12/2016**. Πρόκειται για τον **14ο τόμο** στη σειρά αυτή. Στόχος του θεσμού είναι η ενθάρρυνση της δημιουργικής προσπάθειας και η προβολή των πρωτότυπων εργασιών των φοιτητών του Τμήματος.

Η έκδοση αυτή είναι ψηφιακή και έχει δικό της ISSN. Αναρτάται στην επίσημη ιστοσελίδα του Τμήματος και έτσι, εκτός από τη μείωση της δαπάνης κατά την τρέχουσα περίοδο οικονομικής κρίσης, έχει και μεγαλύτερη προσβασιμότητα. Για το στόχο αυτό, σημαντική ήταν η συμβολή του κ. Ευάγγελου Φλωριά που επιμελήθηκε φέτος την ψηφιακή έκδοση και πέτυχε μια ελκυστική ποιότητα παρουσίασης, ενώ βελτίωσε και την ομοιογένεια των κειμένων.

Η στάθμη των επιλεγμένων εργασιών είναι υψηλή και κάποιες από αυτές έχουν είτε δημοσιευθεί είτε υποβληθεί για δημοσίευση.

Θα θέλαμε να ευχαριστήσουμε τους φοιτητές για το χρόνο που αφιέρωσαν για να παρουσιάσουν τη δουλειά τους στα πλαίσια αυτού του θεσμού και να τους συγχαρούμε για την ποιότητα των εργασιών τους. Ελπίζουμε η διαδικασία αυτή να προσέφερε και στους ίδιους μια εμπειρία που θα τους βοηθήσει στη συνέχεια των σπουδών τους ή της επαγγελματικής τους σταδιοδρομίας.

Η Επιτροπή Ερευνητικών και Αναπτυξιακών Δραστηριοτήτων

Θ. Θεοχάρης (υπεύθυνος έκδοσης), Η. Μανωλάκος

Αθήνα, Ιούνιος 2017

ΠΤΥΧΙΑΚΕΣ ΕΡΓΑΣΙΕΣ

N-Gram Graph Decompression

Despina - Athanasia Pantazi (dpantazi@di.uoa.gr)

ABSTRACT

The current thesis examines the information represented by an n-gram graph. To achieve this task, we searched for all the strings that can be compressed to the same n-gram graph. In order to find these strings, we defined the n-gram graph decompression problem. In addition, we defined the constraint satisfaction problem formulation of the decompression problem, to optimize the latter, and we applied a set of search methods to solve it. We also designed two variable ordering heuristics to improve our time measurements. Finally, we conducted a set of experiments on certain applied search methods to retrieve the initial text from the n-gram graph. We compared the findings from our experiments and we concluded that the best results for short-length texts were achieved when Local Search was performed. For longer-length strings, the weighted degree heuristic was the one that offered the best results.

SUBJECT AREA: Artificial Intelligence

Keywords: n-gram graph, decompression, local search, constraint satisfaction problem, depth first search, breadth first search, heuristics

Supervisors: George Giannakopoulos, Dr (NCSR Demokritos) and Panagiotis Stamatopoulos, Assistant Professor (National and Kapodistrian University of Athens)

1. INTRODUCTION

In the fields of computational linguistics and probability, n-grams are proven a useful approach to represent information with various applications, such as text classification and text summarization. What is most interesting about this tool is that we can search for all the strings that are compressed to an n-gram graph, by defining the n-gram graph decompression problem. By formulating the n-gram graph decompression problem as a constraint satisfaction problem, we are able to retrieve the initial text, which is compressed in an n-gram graph, by applying a set of search algorithms.

According to [12] an n-gram is an n-character slice of a longer string. Although in the literature the term can include any *n* co-occuring characters of a string *s*, in this work an n-gram is simply defined as a substring of a string *s*, where *s* has at least *n* characters. In [3], a statical, language-neutral and generic representation is defined: the n-gram graphs, where n-grams are combined with the powerful concept of the graphs. Using a simple extraction algorithm, a text *T* is compressed to a weighted directed graph, where nodes represent n-grams, the directed edges indicated the connection between a pair of n-grams, and the weight of each edge of the graph indicates the strength of connection each pair of n-grams has. The weight of an edge, between the n-grams *a* and *b,* is the amount of times *b* was found within a distance *Dmax* of *a*.

The n-gram graphs offer richer information than widely used representations. They are not a lossy text representation method, as they use the whole text *T*, by splitting it into n-grams. We use the *JINSECT* toolkit, which is a Java-based toolkit and library that supports the use of n-gram graphs on a whole range of Natural Language Processing applications. The toolkit is a contribution to the *NLP* community, under the *LGPL* license that allows free use in both commercial and non-commercial environments.

By exploring the n-gram graph, we can extract the initial text *T*, which was compressed into an n-gram graph. We compress a text *T* to an n-gram graph *G*, by using the toolkit *JINSECT*, and focus on different algorithmic approaches to retrieve the original text *T,* when we have as input the n-gram graph *G*. The attempt to retrieve the original text is called decompression of the n-gram graph. In this work, we study this decompression process of the n-gram graphs, by executing the search algorithms Local Search (LS), Depth First Search (DFS) and Breadth First Search (BFS). We defined the constraint satisfaction problem (CSP) simulation of the decompression problem, so that the experiments we performed on the above applied search methods can indicate the most optimal search algorithm to complete the decompression procedure. In addition, two variable ordering heuristics were applied to the backtracking algorithms DFS and BFS, in order to maximize the optimization of the results of the decompression problem of the n-gram graphs.

The experimental setting and evaluation of the decompression problem of the n-gram graphs were focused on strings that have fixed length, which is provided by the user. The comparison of the results each search method provides, through its experiments, is based on the time the algorithm requires to find the initial text, and the method cost, which is the amount of methods each algorithm had executed, until the decompression problem was solved.

2. BASIC CONCEPTS AND RELATED WORK

2.1 The N-Gram Graph Representation

In Introduction, an n-gram is described as an n-character slice of a longer string.

Example 2.1.1 Examples of n-grams from the sentence: *This is an example*.
Word unigrams: this, is, an, example
Word bigrams: this is, is an, an example
Character bigrams: th, hi, is, s_, _a, an, ...
Character 4-grams: this, his_, is_a, s_an, ...

The definition of n-gram [3], given a text (viewed as a character sequence) is given below:

Definition 2.1.1 If $n > 0$, $n \in Z$, and c_j is the i-th character of an l-length character sequence $T^l = (c_1, c_2, ..., c_l)$ (our text), then a character **n-gram** $S^n = (s_1, s_2, ..., s_n)$ is a subsequence of length n of:
$T^l \Longleftrightarrow \exists\, i \in [1, l - n + 1] : \forall j \in [1, n] : s_j = c_{i+j-1}$.
We shall indicate the n-gram spanning from c_i to c_k, $k > i$, as $S_{i,k}$, while n-grams of length n will be indicated as S^n .

Definition 2.1.2 Let Q be an alphabet of symbols, L the set of all possible n-length strings of symbols from Q and $S = \{S_1, S_2, ..., S_{l-n+1}\}$ the set of n-grams extracted from a text T^l, made of symbols from Q. We also define a function $sf : S \rightarrow L$, which assigns a n-gram a of the set S to a (unique) string $t = s_1, ..., s_n$ of the set L, if the n-gram a represents the string t, i.e. $a = s_1 s_2 ... s_n$. We will say that a n-gram a is of form of a sting t, if $sf(a) = t$. Furthermore, two distinct n-grams a and b are said to be **of the same form** iff for n-grams a and b we have $sf(a) = sf(b)$.

Example 2.1.2 The 3-grams extracted from the string *Caroline's olive car* are: S_1 = *Car*, S_2 = *aro*, S_3 = *rol*, S_4 = *oli*, S_5 = *lin*, S_6 = *ine*, S_7 = *ne'* , S_8 = *e ' s*, S_9 =*' s_*, S_{10} = *s_o*, S_{11} = *_ol*, S_{12} = *oli*, S_{13} = *liv*, S_{14} = *ive*, S_{15} = *ve_*, S_{16} = *e_c*, S_{17} = *_ca*, S_{18} = *car*. We see that 3-grams S_4 = *oli* and S_{12} = *oli* represent the same string oli, but are found in different positions in the text T^l according to Definition 4.0.2 we shall call these n-gram to be of the same form. However, 3-grams S_1 = *Car* and S_{18} = *car* do not represent the same string; in other words, the n-gram representation we use is *case sensitive*.

The length of a n-gram *n*, is also called the *rank* of the n-gram.

2.2 Related Work

As we mentioned in the Introduction, n-grams are proven a useful approach to represent information with various applications, such as text classification and text summarization. By combining n-grams with the powerful concept of the graphs, we have a statical, language-neutral and generic representation: the n-gram graphs. In the following paragraphs of this subsection, we will refer to previous works, which have used the n-gram graphs in a number of research fields.

In [4], n-gram graphs were used in the context of Sentiment Analysis over Social Media. Through their experimental study, the authors verified that it is a very suitable representation model for this task, as it allows substring matching and, second, it is a language-neutral method that makes no assumptions on the underlying languages. Furthermore, this model exhibits high classification efficiency, as well. It involves a limited number of features that solely depends on the corresponding number of classes. An n-gram graph based method was also used at a methodology for sentiment analysis of figurative language, which applies Word Sense Disambiguation. [6] The n-gram graph representation was used to assign polarity to word senses.

In [5], AutoSummENG system was presented, which is a promising novel automatic method for the evaluation of summarization systems, based on comparing the character n-gram graphs representation of the extracted summaries and a number of model summaries. It was found that statistical information related to co-occurrence of character n-grams seems to provide important information concerning the evaluation process of summary systems. The authors concluded the AutoSummENG method has proved to be a language-neutral, fully automated, context-sensitive method with competitive performance. In [7], a real, multi-document, multilingual news summarization application was developed, named NewSum. This system used the representation of n-gram graphs in a novel manner to perform sentence selection and redundancy removal for the summaries.

In the field of bioinformatics, n-gram graph representation has been a very useful method for inquiring genomic sequence composition. In [8], the authors implemented the n-gram graph approach on short vertebrate and invertebrate constrained genomic sequences of various origins and predicted functionalities. In addition, they were able to efficiently distinguish DNA sequences belonging to the same species.

In this work we study the class of strings a given n-gram graph represents to better acquire insights related to the possible strengths and weaknesses of the representation.

3. PROPOSED APPROACH

The n-gram graph decompression can be formulated as a search problem. Our purpose is to try and find a sequence of steps that will provide the initial text, that was compressed in the n-gram graph. To achieve our goal, we will model the search problem. In addition, we will define the CSP version of the decompression problem.

3.1 Modeling And Solving Search Problems

In this paper, search problems are implemented according to the following assumptions:

1. Every search problem is expressed as a **Problem**. The Problem will try to find a solution by using a tree structure - the **Problem Tree**, and the given search algorithm - the **Search Algorithm**. Each Problem Tree is constructed by **Problem Tree Nodes**, that represent the possible states of our search problem.

2. The Problem needs to decide whether a Problem Tree Node is a solution of the search problem. In addition, it checks whether a problem state is valid and it estimates the distance from a solution. Last but not least, the Problem can get the next states of the problem, based on the current state. There are no other methods the Problem implements, except from the above four.

3. The Search Algorithm accepts a Problem as a parameter and tries to find a solution, by executing the algorithm.

3.2 N-Gram Graph Decompression Problem as a Constraint Satisfaction Problem

Let T^l be an *l*-length (in characters) initial text that can generate the n-gram graph. The *T* was divided in *l* uni-grams. *Let* $G = \{V^G, E^G, L, W\}$ be a n-gram

graph, where V^G is the set of the variables, L is a function that labels each vertex of the graph and W is a function that declares the weight of each edge. Each vertex represents a n-gram of the initial text T. Given the graph G we search for all possible texts T, that can generate the n-gram graph. We expect that $T \in T$. We say that all texts in T are solutions to our problem and we represent such a solution as $S =< n_1, n_2, ..., n_l >$.
We claim that the problem of finding T can be modeled as a constraint satisfaction problem, as follows:
Variables The variables of the problem are the letters n_i of S.
Domain The non-empty domain of each variable is the set of the labels of the unigram graph vertices $L(V)$.
Constraints The set of constraints is the following:
1. Every label $l \in L(V)$ should appear at least once in S.
2. Every label $l \in L(V)$ should appear at most a number of times equal to the weighted degree of the vertex v that is labeled by l. We define as weighted degree of a vertex v the sum of the weights of the edges where v appears.
3. Within a distance D_{win} = 1 of a n-gram there are at most D_{win} neighboring n-grams.
4. For every two neighboring n-grams n_i , n_j of the text S, there is a directed edge $e_i = \{v_x, v_y\}$, where $1 \leq i, j, x, y \leq l$. The amount of times an n-gram n_i is the neighbor of the n-gram n_j is the weight w_i of the edge e_i .
5. There are no other vertices and edges in the graph G but those that are described in the constraints (i) and (ii).
Our goal is the initialization of all the variables while all the above constraints are satisfied.

3.3 Heuristics for the N-Gram Graph Decompression CSP

The two heuristics that will be presented depend on the structure of the n-gram graph decompression problem. The difference of the two heuristics is that by choosing the first one, we will rely on the structure of our problem completely, while choosing the second heuristic will include a randomization factor to the decompression procedure. In the subsection 3.2, we defined the n-gram graph decompression problem as a constraint satisfaction problem, modeled as a search problem, so now we can explain the new heuristics.

3.3.1 Weighted Degree Variable Ordering Heuristic

The first heuristic we will define depends on the weighted degree of the vertices of the n-gram graph. Our goal is to sort the variables of the decompression CSP based on the weighted degree each vertex has.

Example 3.3.1 Assuming we have the n-gram graph *G*, which is initialized by the string *S: queen*. According to subsection 3.2, the variables V^G of the problem are the letters of the *S*, so: *V = {q, u, e, n}*
the weighted degree *wd* of these vertices is: *wd = {q = 1, u = 2, e = 3, n = 1}* which if seen as sorted as: *wd = {q = 1, n = 1, u = 2, e = 3}*

By adding this heuristic to our decompression problem, the vertices will be ordered. The next variable that will be chosen to be branched on will be the one with the minimum weighted degree. When a variable is selected, its weighted degree is reduced by one. If we backtrack while performing the search, the changes that were applied to the variables that are no longer a part of the current solution will have their weighted degree increased.

3.3.2 Probabilistic Variable Ordering Heuristic

The second heuristic we will define is a probabilistic one and will be defined below. Firstly we will describe the basic concepts of the probability theory.

Definition 3.3.1 Probability is the measure of the likelihood that an event will occur.[11] The modern definition starts with a finite or countable set called the sample space, which relates to the set of all possible outcomes in classical sense, denoted by Ω. It is then assumed that for each element $x \in \Omega$, a ”probability” value *f(x),* is attached, which satisfies the following properties:

1. $f(x) \in [0,1]$ for all $x \in \Omega$
2. $\sum_{x\in\Omega} f(x) = 1$

That is, the probability function *f(x)* lies between zero and one for every value of *x* in the sample space *Ω*, and the sum of *f(x)* over all values *x* in the sample space *Ω* is equal to 1. An event is defined as any subset *E* of the sample space *Ω*. The probability of the event *E*, is defined as:

$P(E) = \sum_{x\in E} f(x)$
In our case every vertex *v* of the n-gram graph *G* is an event *V* and will have a probability *P(V).* According to the definition 3.3.1, the probability value *f(v)* will satisfy the above two properties. We define the probability *P(V)* as:

(3.3.1): $P(V) = \frac{1}{w(V)}\theta$

where *θ* is a factor which will be calculated according to the second equation of the 3.3.1 definition, and *wd* is the weighted degree of the particular vertex of the n-gram graph. For the set of the n-gram graph vertices V^G, we have:

(3.3.2): $\sum_{v \in V^G} P(v) = 1 \Longleftrightarrow \sum_{v \in V^G} \frac{1}{wd(v)} \theta = 1 \Longleftrightarrow \theta = \frac{1}{\sum_{v \in V^G} \frac{1}{wd(v)}}$

After calculating the factor *θ*, we will generate one random number per variable, between 0 and 1. The next variable that will be chosen to be branched on will be the one that has its probability number closest to *θ*. When a variable is selected, the weighted degree of it is reduced by one, as we did on the first heuristic in the previous Section. If we backtrack while performing the search, the changes that were applied to the variables that are no longer a part of the current solution will have their weighted degree increased.

Example 3.3.2 Assuming we have the n-gram graph *G*, which is created by the string *S : red*. According to Section 3.2, the variables V^G of the problem are the letters of the *S*, so: *V = {r, e, d}* the weighted degree *wd* of these vertices is: *wd = {r = 1, e = 2, d = 1}* *θ* will be calculated according to equation 3.3.2: $\theta = \frac{1}{\sum_{v \in V^G} \frac{1}{wd(v)}} \Longleftrightarrow \theta = \frac{1}{\frac{1}{1}+\frac{1}{2}+\frac{1}{1}} \Longleftrightarrow \theta = 0.4$

Then, we generate a probability for each variable of the CSP problem, so: *P(V1) = 0.05, P(V2) = 0.23, and P(V3) = 0.68*

These probabilities will be sorted, and we will have the following order: *sortedProbabilities = {P(V2), P(V3), P(V1)}.*

According to the probabilistic heuristic, the next variable that will be chosen to be branched on will be the *V2*. The weighted degree of it will be reduced by one.

4. EXPERIMENTAL SETTING AND EVALUATION

This Section presents the experimental results and comparisons on a number of synthetic datasets. All experiments are conducted on a processor at 2.3 GHz with 4 GB memory. The experiments are conducted so that we can compare the time measurements and the method cost measurements of the search algorithms we implemented to solve the n-gram graph decompression problem, which are Local Search, Depth First Search, and Breadth First Search. After getting a string as an input, we compress it into an n-gram graph, by using the toolkit *JINSECT*. We ask the user to choose the search method we will execute in order to decompress the n-gram graph, and we get a file as an output, which includes the decompressed string, the execution time, the solutions we found and the method cost.

4.1 Datasets

In the main source file, the defined String *CHARS* contains the following 100 characters:
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz!@#
Each time a search is executed, the user is asked to provide:
1. the number of the strings that will be generated
2. the length of each string
3. the probability of repetition for the current set of strings that will be produced.

Furthermore, the user can choose to load their own datasets, by providing the name of their file, at the beginning of the execution of the program. In this thesis, the probability for all the following tests is 0%. As a result, all the characters each generated string will have cannot be repeated. In addition, we have an upper limit for the amount of executions while performing the backtracking algorithms. This limit is the 100 million attempts to check if the string we have constructed by that time is the solution of the algorithm. For comparability reasons, the datasets that are used for all the following experiments are the same.

4.2 Experiments

The following table gives us a general view of all the search methods we used to decompress the n-gram graphs.

	length of text [chars]			
	4	5	6	7
Local Search	**0,58**	2,62	26,12	454,75
DFS simple	11.841,68	-	-	-
DFS with CSP	4,15	19,75	221,24	3.222,16
DFS with CSP & WD heuristic	0,59	**1,2**	**7,19**	**71,61**
DFS with CSP & probabilistic heuristic	1,04	3,81	22,09	303,9
BFS simple	-	-	-	-
BFS with CSP	7,51	63,08	905,6	17.343
BFS with CSP & WD heuristic	0,98	4,56	46,41	1.940,1
BFS with CSP & probabilistic heuristic	1,54	8,77	100,75	5.254,52
Best time	**0,58**	**1,2**	**7,19**	**71,61**

Table 1: General time measurement comparisons Time in milliseconds, the total strings were 1000 for every algorithm measurement. The dash symbol (-) indicates no results could be provided.

It is shown that the Local Search is a quick search method for the decompression of n-gram graphs that have strings consisted by a few characters. As you can notice, the simple BFS diagrams are not included, because an ”out of memory” error had occurred during our executions. This behavior is expected, due to the logic of this algorithm, when dealing with strings that do not have multiple appearances of their characters. The simple DFS diagrams for the 5, 6 and 7-character length are not included either, as the upper limit for the amount of executions while performing the backtracking algorithms (100 million attempts) was reached. When we want to decompress longer strings, we should prefer the DFS or BFS algorithm for the CSP simulation of our problem. The heuristics improve our time measurements, especially the weighted degree heuristic.

5. CONCLUSION

In this thesis, we have implemented and performed experiments on a set of search algorithms, in order to solve the decompression problem of the n-gram graphs. We defined the constraint satisfaction problem formulation of the decompression problem, in order to optimize it, and we designed two variable ordering heuristics to improve our time measurements. The experimental results presented judged one class of character strings to be decompressed, these which do not have multiple character appearances. The best results for short-length text were achieved when Local Search was performed. For longer-length strings, the weighted degree heuristic was the one that offered the best results.

The results presented above are encouraging and can be improved. The parallelization of the search algorithms we analyzed could offer betters time measurements. Furthermore, new variable ordering heuristics and value ordering heuristics can be designed and implemented, so that we can determine which of them can provide better experimental results for the decompression problem. Future work on the decompression problem of the n-gram graphs can include experiments focused on strings that have multiple appearances of the same character.

REFERENCES

[1] S. Russell, P. Norvig. (2003), Artificial Intelligence - A modern approach, Second Edition, Prentice Hall, pages 179-200.
[2] Kenneth H Rosen. (2011), Discrete mathematics and its applications, Seventh Edition, McGraw-Hill New York.
[3] George Giannakopoulos. (2009), Automatic summarization from multiple documents, Citeseer.

[4] F.Aisopos, G.Papadakis, T.Varvarigou (2011), Sentiment Analysis of Social Media Content Using N-Gram Graphs, Proceedings of the 3rd ACM SIGMM international workshop on Social media.
[5] G. Giannakopoulos, V. Karkaletsis, G. Vouros, P, Stamatopoulos (2008), Summarization System Evaluation Revisited: N-Gram Graphs, ACM Trans. Speech Lang. Process.
[6] V. Rentoumi, G. Giannakopoulos, V. Karkaletsis, G. Vouros (2009), Sentiment Analysis of Figurative Language using a Word Sense Disambiguation Approach, Borovets, Bulgaria
[7] G. Giannakopoulos, G. Kioumourtzis, V. Karkaletsis (2014), NewSum: "N-Gram Graph"-Based Summarization in the Real World, Innovative Document Summarization Techniques: Revolutionizing Knowledge Understanding, IGI
[8] D. Polychronopoulos, A. Krithara, C. Nikolaou, G. Paliouras, Y. Almirantis, G. Giannakopoulos (2014), Analysis and Classification of Constrained DNA Elements with N-gram Graphs and Genomic Signatures, Algorithms for Computational Biology, Springer International Publishing
[9] D. Poole, A. Mackworth. (2010), Artificial Intelligence: Foundations of Computational Agents, Cambridge University Press, 4.8
[10] F. Rossi, P. van Beek, T.Walsh (2006), Handbook of Constraint Programming, Elsevier B.V., 4.6
[11] G and C Merriam, Webster's Revised Unabridged Dictionary (1913)
[12] William B Cavnar, John M Trenkle, et al. N-gram-based text cat- egorization. Ann Arbor MI, 48113(2):161–175, 1994.
[13] Faidra Monachou, Designed and implemented two n-gram graph decompression algorithms in Java using the JInsect Toolkit, 2013, Personal communication with supervisor.
[14] A. Auger, B. Doerr. (2011) Theory of Randomized Search Heuristics: Foundations and Recent Developments, preface [15] K. Potter (2006) Methods for Presenting Statistical Information: The Box Plot.

Author Profiling in Social Media using Topic Modeling methods

Alexandros F. Zeakis (sdi1200038@di.uoa.gr, alzeakis@gmail.com)

ABSTRACT

In Author Profiling Task, researchers, given a number of texts, try to find the characteristics of the author, e.g. Age and Gender, based on stylistic- and content-based features. In this thesis we tried to solve the Author Profiling Task utilizing topic modeling methods, such as *Latent Semantic Indexing* and, mainly, *Latent Dirichlet Allocation.* To this end, we represented each document as a mixture of topics and then used this latent representation as input features for known classification algorithms, such as *Support Vector Machine*, to create our predictive system. To be noted, our approach was a part of the solution that was submitted to the 4th Author Profiling Task at PAN 2016.

We used two corpora for this task, one based on blogs and one on tweets, while all documents were preprocessed by known Natural Language Processing (NLP) methods. The development of the system consists of phases, where in each one specific parameters of the model were optimized and finalized. Experimental results show that topic modeling can be used for authorship age and, mainly, gender prediction. In summary, the proposed methodology demonstrates that latent topics make for good descriptors regarding age and gender of the authors and also provides us with new means to explore the discussion themes among age groups and genders.

Keywords: Topic Modeling, Latent Dirichlet Allocation, Author Profiling, Information Retrieval, Latent Semantic Indexing

Advisors
Anastasia Krithara, Associate Researcher (N.C.S.R. Demokritos), Georgios Paliouras, Research Director (N.C.S.R. Demokritos), Panagiotis Stamatopoulos, Assistant Professor

1. INTRODUCTION

Automatic Authorship Identification (AAI) exists for almost 120 years. Mendenhall [10] was the first to examine works of Bacon, Shakespeare and Marlowe aiming to detect quantitative stylistic differences using word length. Since then, things have changed rapidly due to the development of technology [16].

There are three major fields in AAI: Authorship Attribution, Author Identification and Author Profiling. In the first two, the goal is to recognize the author from a set of authors, while in Author Profiling, the goal is to find specific characteristics of the author, based on stylistic- or content-based features.

The Author Profiling task is a yet-unsolved problem, due to its difficulty. It has been studied by many researchers [1;2;8;11;12;13;14] and, while some show great progress and good results, it still has many unexplored areas and room for improvement. This was one of the main incentives for our research. That is, to create a novel system of prediction, based on the topics of discussion between different age groups and genders.

In our approach, we concatenated each author's texts in one big document and then preprocessed that document with known natural language processing methods, such as the removal of punctuation or common words. Then, using Latent Dirichlet Allocation, we extracted the latent topic distribution for each document and used it as input features in a Support Vector Machine classifier.

In this thesis we present the necessary steps of developing the aforementioned system along with interesting findings that were discovered during its creation.

2. PROPOSED APPROACH

The purpose of this chapter is to analyze the approach that we followed to solve the Author Profiling Task, while participating in PAN 2016. This chapter follows every step of the proposed methodology, explaining the importance of it and justifying every choice that we made.

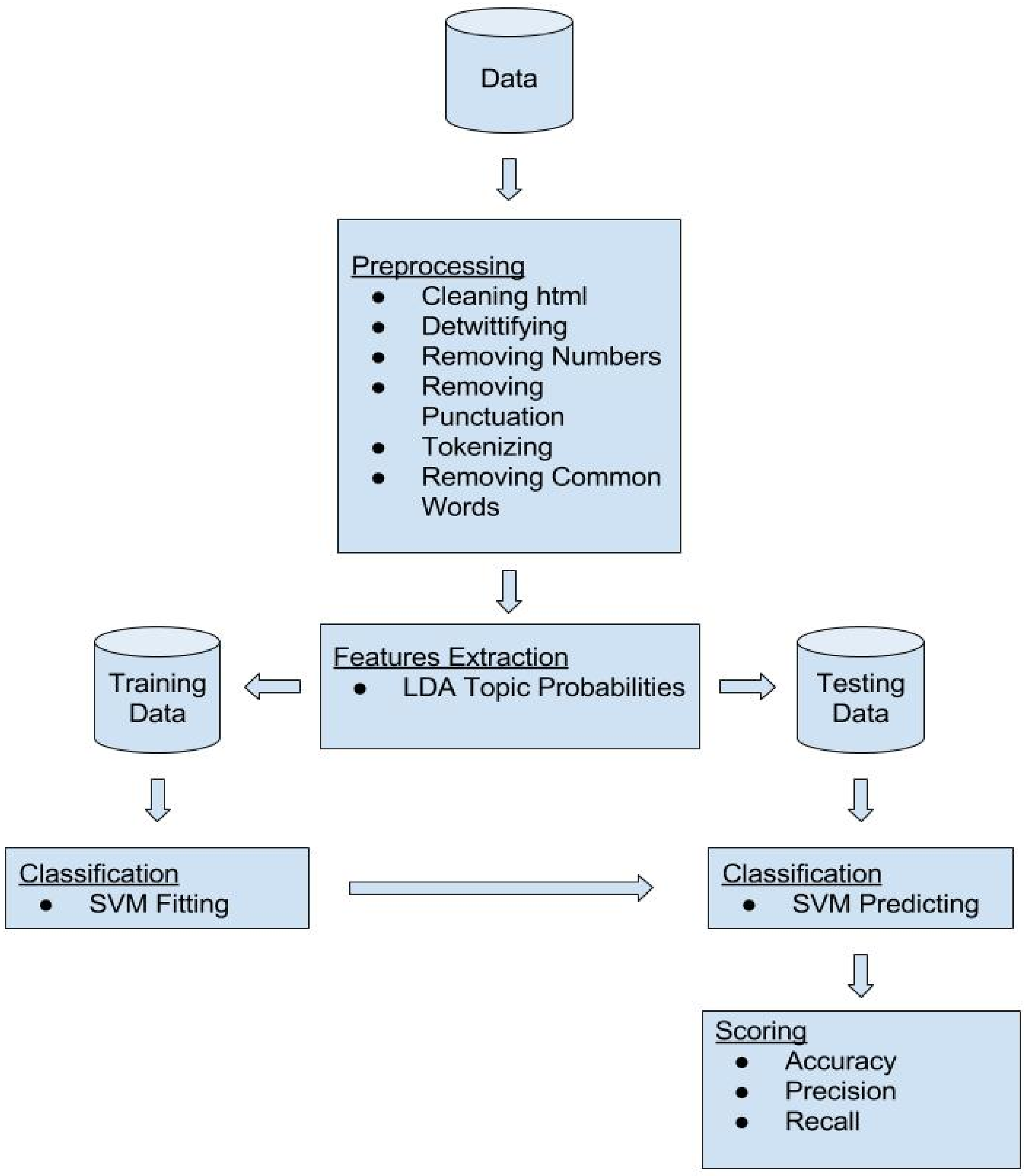


Figure 1: The workflow of the proposed approach

2.1 Preprocessing

Preprocessing is essential to clean data from irrelevant information and transforming them into a form appropriate for feature extraction. These steps include wide-accepted methods, such as cleaning html, removing numbers and common words, tokenizing. We handpicked also methods specifically for this task, which is detwittifying (cleaning texts from twitter elements, such as @reply, hashtags, links) and removing punctuation, because in this task, finding topics does not require punctuation.

2.2 Feature Extraction

While most of the approaches use features that are proven effective, such as n-grams, we wanted to experiment on something different, since our approach is an extension of [7]. The main idea was to get the per-document topic probability distribution and use that to train the classifier.

We used two known methods for Topic Modelling [5]: Latent Dirichlet Allocation (LDA) and Latent Semantic Indexing (LSI). There are enough differences between LDA and LSI, but one could claim that the main one is that LDA is probabilistic and is based on Dirichlet Distributions, while LSI is deterministic utilizing linear algebra methods.

Latent Semantic Indexing is used more frequently as a way to reduce data dimensions, because the known linear algebra method Singular Value Decomposition (SVD) is used. In Information Retrieval, though, is used to create topic models [9], as it can be used to detect relations between words. There are two phases in LSI: the first one is SVD, while the second is rank lowering, meaning that only some of the transformed data are preserved in order to reconstruct the original matrix. In our approach we used the library Gensim[1].

On the other hand, the basic idea behind Latent Dirichlet Allocation, as it was described by D.Blei, A.Ng and M.Jordan in [4], is as follows: Each document can be represented as a mixture of words. If we can find distinct groups of words, based on the frequency of the words in them, then each word could be represented as a mixture of these groups, called topics. As a result, each document can be represented as a mixture of topics. In our approach we used two different implementations of LDA. The first one[2] implements the Online Variational Bayes (VB) algorithm for LDA [3] and the second one[3] creates the

1 https://radimrehurek.com/gensim/models/lsimodel.html

2 https://radimrehurek.com/gensim/models/ldamodel.html

3 http://mallet.cs.umass.edu/

generative model through the implementation of Collapsed Gibbs Sampling Algorithm [6].

2.3 **Classification**

Classifiers are one of the most crucial parts of the problem and selection task is very important. Fortunately, Support Vector Machines (SVM) are a very powerful category of classifiers and can be used in a wide variety of problems, so it is only natural that we preferred them. We also used Naive Bayes, but only for cross-reference results and in the first few steps of our implementation. It became clear later that we should use only SVM, because the results of SVM were much better than the ones from NB.

Support Vector Machines are supervised learning models. They can work efficiently with high-dimensional data, while avoiding the curse of dimensionality. (Curse of dimensionality refers to the phenomenon, where we examine data in a high-dimensional space, which are inevitably scattered, and thus impeding the task of either classification or clustering.) The trick of achieving that is using a subset of the training data, known as support vectors.

3. EXPERIMENTS & RESULTS

This chapter consists of two sections: the first one (Experimental Setup) is about the corpora that were used and the metrics used to evaluate the classifier, while the second one (Results) describes in detail every phase of the development.

3.1 Experimental Setup

3.1.1 Datasets

The quality of the data is very important for the quality of the outcome. We used two datasets during the development of the classifier. The first one [15] was based on blogs and consisted of 19320 authors, where the length and number of texts of each author was different. In the next Figure it is clear that, in

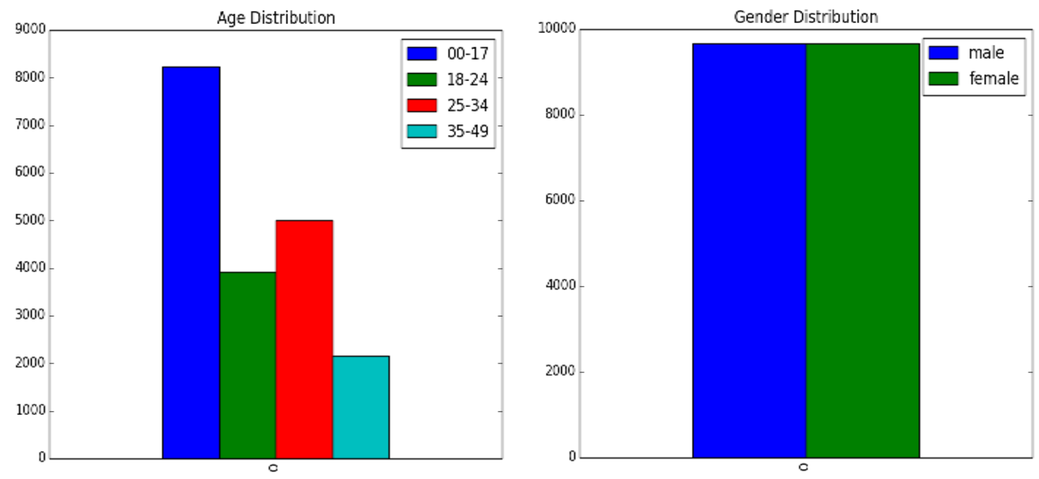
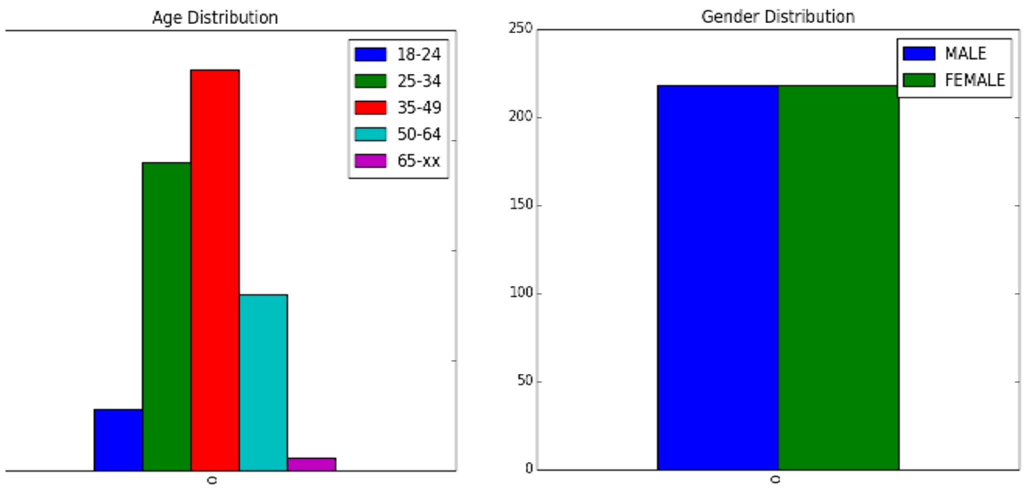


Figure 2: Age (Left) & Gender (Right) Distribution on blog corpus

Gender, the classes are balanced, while in Age some classes are skewed.

The second corpus was the dataset for PAN 2016, based on tweets and consisted of 436 authors. In contrast to the previous corpus, each text has maximum 140 characters, but each author can have many texts. Gender classes are again balanced, while in Age some classes are under-represented, such as '65-xx'.



Figure

eft) & Gender (Right) Distribution on PAN 2016 corpus

3.1.2 Evaluation Measures

The metrics are necessary to measure the performance of the classifier. Assuming that there are two classes, a positive one and a negative one, there are four groups of predictions: true positive (tp), false positive (fp), false negative (fn) and true negative (tn). When a prediction is characterized as true, it means that the current prediction was correct and when is characterized as false, it means that it was incorrect.

When these four categories are defined we can use them to measure the performance with these metrics:

Measure	Formula	Description
Accuracy	$\frac{tp+tn}{tp+fn+fp+tn}$	How many instances were correctly predicted compared to all data
Precision	$\frac{tp}{tp+fp}$	Fraction of positive predicted samples that were actually positive
Recall	$\frac{tp}{tp+fn}$	Fraction of positive predicted samples over all positive samples

Table 1: Measures used for evaluation

In all three metrics, the higher the classifier achieves, the better.

3.2 Results

In this section we are going to present the results of our experiments, based on the approach we developed in the previous chapter. Each subsection in this section follows a major phase during the development of the classifier. Each phase is about a critical decision that affects the rest of the development, such as the size/number of chunks of text or the number of topics. The phases are in chronological order.

3.2.1 Phase 1: Size of Text

At this phase, due to the unavailability of the PAN corpus, we used the blog corpus. Our first approach was to use chunks of text for each author, but there was no constant number of chunks or size and in LDA, 40 topics were used. To be noted, these tests were performed on the training data, since if the performance based on the training data couldn't be improved, then good results when classifying on test data would be unreachable. Accuracy in predictions in Age were quite low (less than 60%) and only one label was predicted, so a suggested fix could be to concatenate all texts of each author, thus transitioning from Instance-based Classification to Profile-based Classification.

While there was in improvement in the new approach (more than 90%), when predicting on testing data than number fell to ~65%. Nevertheless, between SVM and NB classification algorithms, the best performance was shown by SVM. In addition to that, concatenation of the texts proved much more efficient than chunks of text, so it was kept for the next phase.

3.2.2 Phase 2: Number of Topics

Since in that phase we switched to the PAN corpus, we had to recalibrate the number of topics. We ran many experiments using all possible parameters. From now on, when the term *triplet* is mentioned, it is going to be in the form of {no_topics, kernel_of_SVM, C_of_SVM}. After consecutive executions on the best parameters, we discovered that the Gensim Library was quite unreliable, as the same parameters and same portion of data could lead to different results. The same could be said for Mallet, but Mallet results differed very little between executions, while the deviance of results in Gensim led to the decision to use only Mallet for the next experiments. The best triplet for both Age and Gender was {20, linear, 100}.

3.2.3 Phase 3: Stratify and More Authors

For the next phase, we wanted to improve the metrics and we started thinking ways to do that. Among many ideas, the stratified splitting of data came up. In stratified splitting, proportionally equal amounts of each class are used in both training and testing phase. Thus, the higher occurrence or absence of a class in the training data should be dealt with and not affect the performance of the classifier.

As a next step, the number of authors was increased in order to test the classifier on bigger data. Results were not that good (Age Accuracy less than 40%), but both steps were necessary for the next and final phase.

3.2.4 Phase 4: Final Results

In this final phase, we present the final results with the best parameters (triplets), that were found after many experiments.

There are classes in Age that were under-represented, such as 65-xx. In the Topic Distribution diagrams, though, it could be seen that every class had a chance to dominate in some topics. The same could be said about each topic's most characteristic words, as most words could easily lead to the writer's age class. While there were topics that were distinctively true, most of them were incorrect and the same behaviour occured in Gender. The final results are presented in the following table:

	{20, linear, 100}	{20, linear, 100}
Accuracy	0.3065	0.6230
Precision	[0.25, 0.3913, 0.2917, 0.1818, 0]	[0.64, 0.6111]
Recall	[0.2, 0.45, 0.3043, 0.1538, 0]	[0.5333, 0.7097]
Fbeta	[0.2222, 0.4186, 0.2979, 0.1667, 0]	[0.5818, 0.6567]

Table 2: Final Results on Age (Left) & Gender (Right)

4. CONCLUSIONS & FUTURE WORK

The goal of this thesis was to prove, that given the text of an author, his/her Age and Gender could be found, based on stylistic- and content-based features. After many experiments, the following conclusions were produced:

- Concatenated texts produce better results than chunks of texts. Yet, neither the chunks of texts that were used in this approach had constant size, nor each author had the same amount of chunks. So, we can not claim that in every problem concatenating texts can be the solution.
- Solely based on figures, LDA produced better results than LSI. So, it is clear that LSI is not an algorithm appropriate for this kind of problem.
- The implementation of LDA in Mallet (Gibbs sampling) was more stable than the one in Gensim (online Variational Bayes), meaning that the results of LDA-Mallet did not have a wide deviance between executions.
- From the two classification algorithms that we used, *Support Vector Machine* and *Naive Bayes*, it was clear after some experiments, that SVM was more appropriate for this task.
- Classes were better separated by a linear classifier, as the best triplets of our executions included kernel 'linear'.
- Based on the final results, the problem of Gender was handled better than the problem of Age. One explanation could be that in Gender there are only two classes and they were also balanced, while in Age, there were five classes, which were skewed and that added an extra difficulty in the task.

The results, though, in general, were not good and that implies that further work is required in order to improve the results of our system and the quality of the predictions. Based on the related work and the choices that we made, the following suggestions could lead to an improvement.

- Instead of concatenated texts, we could try again chunks of texts, this time with a specific number of chunks per author or a specific size for each chunk. Thus, authors with few/small texts could have the same representation as the others.
- Lemmatizer is generally a good preprocessing step, although we did not use it in the final approach. WordNet Lemmatizer performed poorly, as words expected to be lemmatized, were not. Observing the final topic words, a lemmatizer (or stemmer) is required in order to improve further the results.
- Language filtering is necessary, as there are topics, as seen in the topic words of the final results, whose entire top-words are non-english.
- While using only the LDA output as features was producing poor results, if combined with some other features (e.g. n-grams), it could lead to improvement in the results.
- While LDA was the best topic modeling algorithm of those we tried, we could still try others, such as Hierarchical Dirichlet Process (HDP).

REFERENCES

[1] Argamon, Shlomo, et al. "Mining the blogosphere: Age, gender and the varieties of self-expression." *First Monday* 12.9 (2007).

[2] Argamon, Shlomo, et al. "Gender, genre, and writing style in formal written texts." *TEXT-THE HAGUE THEN AMSTERDAM THEN BERLIN-* 23.3 (2003): 321-346.

[3] Blei, David, and M. Hoffman. "Online Learning for Latent Dirichlet Allocation." *Neural Information Processing Systems*. 2010.

[4] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent dirichlet allocation." *Journal of machine Learning research* 3.Jan (2003): 993-1022.

[5] Chang, Jonathan, et al. "Reading tea leaves: How humans interpret topic models." *Advances in neural information processing systems*. 2009.

[6] Griffiths, Tom. "Gibbs sampling in the generative model of latent dirichlet allocation." (2002).

[7] Grivas, Andreas, Anastasia Krithara, and George Giannakopoulos. "Author profiling using stylometric and structura feature groupings—notebook for pan at clef 2015." *CLEF 2015 Evaluation Labs and Workshop–Working Notes Papers*.

[8] Koppel, Moshe, Shlomo Argamon, and Anat Rachel Shimoni. "Automatically categorizing written texts by author gender." *Literary and Linguistic Computing* 17.4 (2002): 401-412.

[9] Laham, T. K. L. D., and Peter Foltz. "Learning human-like knowledge by singular value decomposition: A progress report." *Advances in Neural Information Processing Systems 10: Proceedings of the 1997 Conference*. Vol. 10. MIT Press, 1998.

[10] Mendenhall, Thomas Corwin. "The characteristic curves of composition." *Science* (1887): 237-249.

[11] Nguyen, Dong-Phuong, et al. "" How old do you think I am?" A study of language and age in Twitter." (2013).

[12] Rangel, Francisco, et al. "Overview of the 3rd Author Profiling Task at PAN 2015." *CLEF*. 2015.

[13] Rangel, Francisco, et al. "Overview of the 2nd author profiling task at pan 2014." *CEUR Workshop Proceedings*. Vol. 1180. CEUR Workshop Proceedings, 2014.

[14] Rangel, Francisco, et al. "Overview of the author profiling task at pan 2013." *CLEF Conference on Multilingual and Multimodal Information Access Evaluation*. CELCT, 2013.

[15] Schler, Jonathan, et al. "Effects of Age and Gender on Blogging." *AAAI Spring Symposium: Computational Approaches to Analyzing Weblogs*. Vol. 6. 2006.

[16] Stamatatos, Efstathios. "A survey of modern authorship attribution methods." *Journal of the American Society for information Science and Technology* 60.3 (2009): 538-556.

Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών,
Πανεπιστημιούπολη, 15784 Αθήνα

ISSN 1792-8826