A Language of Life: Characterizing People using Cell Phone Tracks

Alexy Khrabrov and George Cybenko
{*alexy,gvc*}@dartmouth.edu
Thayer School of Engineering
Dartmouth College
8000 Cummings Hall
Hanover, NH 03755

***Abstract*—Mobile devices can produce continuous streams of data which are often specific to the person carrying them. We show that cell phone tracks from the MIT Reality dataset can be used to reliably characterize individual people. This is done by treating each person's data as a separate *language* by building a standard n-gram language model for each "author." We then compute the perplexities of an unlabelled sample as based on each person's language model. The sample is assigned to the user yielding the lowest perplexity score. This technique achieves 85% precision and can also be used for clustering. We also show how language models can also be used for predicting movement and propose metrics to measure the accuracy of the predictions. Finally, we develop an alternative method for identifying individuals by counting the subsequences in a sample which are unique to their authors. This is done by building a generalized suffix tree of the training set and counting each subsequence from a sample which is unique for some person as evidence towards identifying that person as the author. We present the identification and prediction as a part of a HUMBLE human behavior modelling framework, outline general modelling goals, and show how our methods help. Our results suggest that people's medium-scale movement behavioral patterns, at the granularity of cell tower footprints, can be used to characterize individuals.**

I. Introduction

With the widespread adoption of mobile telecommunications devices and the networks to support them, there now exist continuous data streams emitted by many people throughout the day. Most notably, when mobile phones move through a cellular network, the IDs of the cell towers tracking the phone can be recorded as a timestamped integer sequence. Similar ID sequences can be produced by a smart card moving near readers, as a series of reader IDs – recording, for example, the trajectory of an employee's badge through a secure building, a passenger through a smart transportation system, and so on.

With many such trajectories available over time, the question arises of the uniqueness, regularity, and predictability of people's movements as based on those trajectories. We address the problem of identifying individuals based solely on a subsequence of cell tower observation records. We develop two methods of identification from a sensor data stream, and also show how one model can be used for prediction. We place these methods in a larger context of human behavior modelling, and propose a framework for reasoning about such modelling, with identification and prediction forming the basis. Future work, naturally completing the framework, is discussed.

We present identification and prediction techniques for behaviors captured in the MIT Reality dataset [2]. In that dataset, about 100 subjects, MIT students and faculty, were given cell phones with tracking software, recording, among other data, their cell locations and times of calls.

II. Previous Work

Author identification from symbol sequences was treated before in the literary authorship context – Markov models were applied to character alphabets, with good results [10]. Suffix trees were used for finding member-unique subsequences in a collection of sequences in bioinformatics, for selecting primers [8]. Corpus fertility was introduced by the first author and others in [13]. Metrics for similarity between sequences using suffix trees are proposed in [16]. Gonzalez et al. [5] show that people are generally fixed in their way and exhibit patterns of mobility in cell phone network.

III. Human Behavior Modelling Framework

This work is a part of the Human Behavior Modelling project, (HUMBLE), developed at Thayer School of Engineering, Dartmouth College [17], for integrating and generalizing various techniques for modelling, representing and using human behaviors as inferred from sensor data. The main purposes of human behavioral modelling in this framework are as follows:

- *Characterize* - The ability to identify unique individuals or classify them robustly.
- *Identify Change* - The ability to detect significant changes in the behavior of an individual.
- *Predict* - The ability to predict future behaviors based on observed past behaviors.

In this paper, we focus on the first challenge (characterization), briefly explore the third (prediction), and outline our plans for the second (change detection). Characterization can be seen as a classification problem, and we start with it in a very strict sense – as identification of a class of one. Specifically, given a sample sequence produced by an unidentified person from our set of subjects, we seek to identify the person who produced it.

TABLE I
AVERAGE DURATIONS OF THE SAMPLES, FOR GIVEN SAMPLE SIZES.

sequence length	10	20	50	100
duration	41 min	81 min	3.4 hours	6.8 hours

It takes less than an hour to gather a characteristic sequence from an average person.

IV. CELL PHONE TRACKING

A. Background on MIT Reality Data

The *cellspan* table of the MIT Reality data tracks movements of each person through the cellular network [2]. Every time a new cell is entered, its ID is recorded, along with the time stamp of the transition. it is implicitly assumed in the study that individual cell phones were used solely by one individual throughout the study, so that cell phones and people are associated uniquely.

We have a total of 2.5 million transition for all 89 subjects who have data; the shortest person-corpus consists of only 846 observations, while the longest has 79,221. The histogram of corpora sizes is shown in Figure 1.

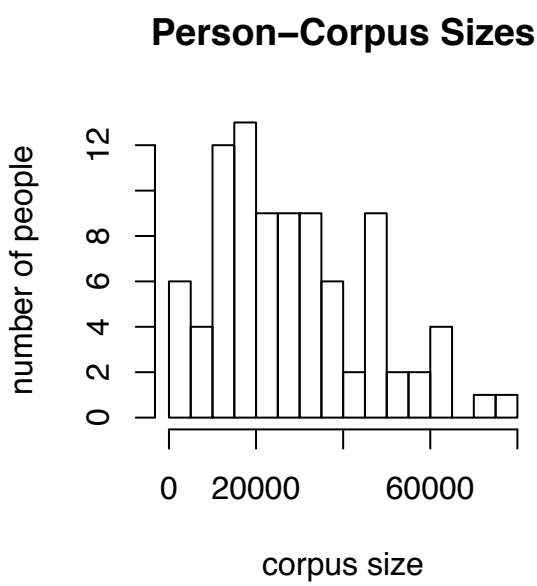

Fig. 1. Person-corpus sizes distribution for MIT Reality data. 89 subjects have 2.5 million cell transitions total, with corpus sizes ranging from 846 to 79,221. Most people generate large tracks of their movements.

The average step – time between the current and the next cell ID reading is recorded – is about 4 minutes. We randomly pick contiguous "future" samples of fixed length to identify the people who produced them. The average time span for the fixed sample lengths we use are shown in Table I.

B. Calling Time Averages

A standard way to model irregular time series data is to reduce it to standard period metrics which can then be classified. For call data, we considered various time characteristics first – the duration of a call, the time of day the call is made, the interval between calls, etc., and computed weekly averages of those features for each person. We then fed those people-weeks of timing feature vectors into a support vector machine (SVM) classifier, identifying the class – a single person ID – of a de-labelled weekly vector. The call timing features used are shown in Table II. The results are shown in Figure 2. There are about 10 people whom we can identify better than 50% of the time, a middle group which can be identified with some accuracy, and the remaining half, not shown in the graph, who cannot be identified at all using this method. Obviously, time patterns are not enough – we need to consider sequence data, and find a better identification method.

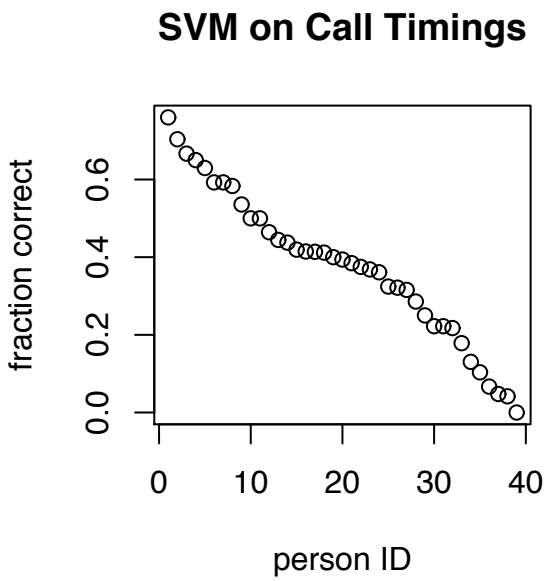

Fig. 2. Identifying people with multi-class SVM on their weekly averages of call timing data. We don't get most of the people above 50%, and such identification doesn't work far into the future.

V. MODELLING SENSOR DATA AS A LANGUAGE

The cell ID sensor data can be reduced to a sequence of integers (with optional timestamps). In the MIT Reality dataset, the integers are the IDs of the cell towers tracking a phone – and a person's – movements. We propose treating each person's sensor data stream as a separate *language*, and model it with n-gram Markov chains. Below we review n-gram language modelling, and then compare sensor corpora to each other and real natural language corpora of English and Russian, in terms of the fertility concept.

Now we have, for N_P people, N_P corpora, each representing its own language. The words in these languages are sensor readings, in our case cell tower IDs. The utterances are movements, leaving a trace of the IDs behind. Currently we

TABLE II
CALL TIMING FEATURES

feature name	*description*
n	total number of calls
nz	number of non-zero length calls
tcl	total calling time
(nz.)duration.mean	average duration of a call
(nz.)duration.sd	standard deviation of call duration
start.mean	average call start day time in seconds from midnight
start.sd	standard deviation of call start time from midnight
interstart.mean	average time between call starts
interstart.sd	standard deviation of inter-start times

Aggregated weekly per person, used for SVM-based identification. For mean and sd features we also compared median and *mad* robust estimators. The nz versions take into account only non-zero length calls, which also excludes SMS messages. We also separated business days from weekends, without much difference in prediction quality.

treat the whole history as a single sentence. Later variations may treat sensor IDs as letters, and try to build words on top of them, and then group those words into sentences.

The original n-gram model, a Markov chain, was developed by Andrei Markov for an analysis of the text of Pushkin's "Eugene Onegin" [15]. It naturally applies to sensor data when it is treated as a language. A bigram corresponds to a transition from one cell to another, and an n-gram in general is then a segment of a user's trajectory. The question is to estimate the probability of a sentence, a sequence of words, from a corpus. The general way to do it is by developing a language model Φ, a "grammar" generating possible sentences, and use Equation 1 to compute the probability.

$$P(\mathbf{W}) = \prod_{i=1}^{n} P(w_i|\Phi(w_1,\ldots,w_{i-1})) \quad (1)$$

$$\Phi_i = S(w_{i-n+1},\ldots,w_i) \quad (2)$$

The most common way of building Φ is to count the occurrences of n consecutive words, the n-grams, up to some order N, and use such counts in a maximum-likelihood estimate, with various kinds of smoothing S to redistribute some probability mass from observed n-grams to those unseen (so the product never zeroes out). While the state of the art smoothing for natural languages is Kneser-Ney smoothing [1], [6], it turns out the counts-of-counts used by Kneser-Ney are too sparse for our sensor data, and we use Witten-Bell smoothing instead [20].

In speech and text processing, corpora atoms are either phonemes or words, with n-grams then corresponding to words or phrases (collocations), respectively [9], [14]. We use the word/phrase terminology in our analogies with natural language processing. An n-gram language model represents a corpus as a Markov chain of order $n-1$.

A. Cell ID Fertility

Corpus fertility is a metric intoduced in [13] to quantify how much of the corpus is "enough" for modelling phenomena of interest. The idea is, we can stop collecting corpora when

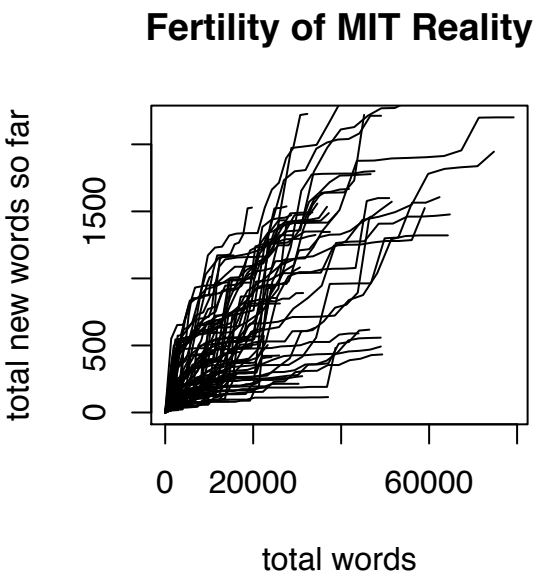

Fig. 3. Fertility of 89 people-corpora of sensor data. The curves are not showing saturation seen in the natural languages corpora, and sharp increases mean a new area is covered.

the number of new words (unigrams) or generally n-grams encountered per unit of new raw text becomes low. The original methodology was applied to Russian and English corpora of classic texts and press articles. The corpus is split into chunks of equal size, for example 1,000 words, and these chunks are "walked" in sequence, either original or permuted. While walking, we count, for each new chunk, how many new words are encountered. The resulting curve shows the fertility of the corpus, typically rapidly growing in the beginning and asymptotically power-law in the end.

It is interesting to compare our sensor data corpus with a natural language such as English or Russian. For an idea, we show some fertility graphs below. On Figure 3, all 89 people's fertility is shown; Figure 4 shows the fertility of a 2.4 billion word English press corpus, and Figure 5 shows the fertility of a 1 billion word Russian press corpus, both collected in 2007 from the contemporary major periodicals such as the New York Times and the Kommersant.

Natural languages show the same power law slow growth. On the other hand, the cell ID sensor data corpora, while generally adhering to it, show higher variability. There seems to be enough novel travel to inject new sets of IDs into most people's streams. Such jumps in fertility may be good separators of the segments to model individually.

VI. N-GRAM RANKING FOR SENSOR DATA STREAMS

We use SRILM [18], the industry standard toolkit for language modelling by Andreas Stolcke. SRILM implements various smoothing methods needed to redistribute some prob-

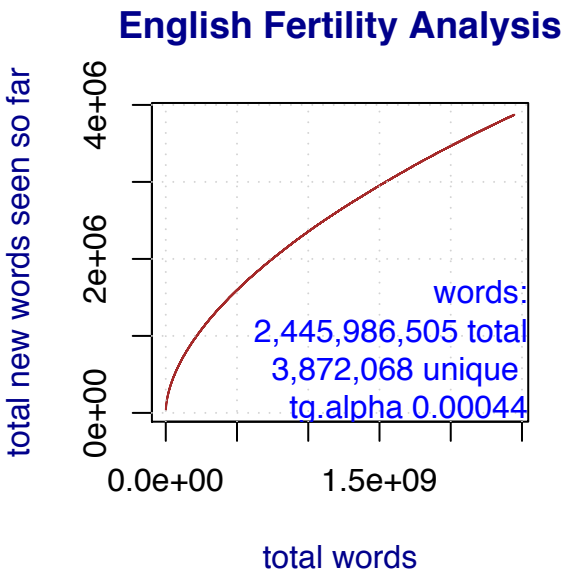

Fig. 4. English press corpus fertility. Out of 2.4 billion words, about 3.9 million are unique; we walk the corpus in 1K word increments, counting the uniques per chunk, accumulating total uniques count on the Y axis. Shown at 0.5 billion ticks on X. The tg.alpha shows the angle of the regression line fitted to the log-log plot of the data – basically, we're getting 0.4 new words per each 1,000 words of corpus in the end, usually due to OCR errors, but sometimes neologisms.

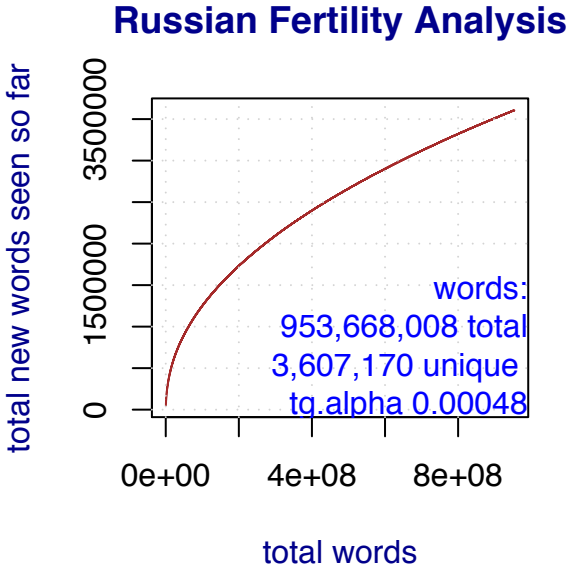

Fig. 5. Russian press corpus fertility. 1 billion word corpus produces about 3.6 million unique words; see English fertility caption above for the fertility walk methodology. Shown at 0.2 billion ticks on X.

TABLE III
5-GRAM IDENTIFICATION OF MIT REALITY DATA

batch size \ *sample length*	*10*	*20*	*50*	*100*
1	35	46	52	53
2	25	41	49	53
3	44	50	53	54
5	44	52	54	54
10	41	54	54	54
20	49	54	54	54
50	47	54	54	54
100	47	54	54	54
1000	51	54	54	54

Training on 63 subjects up to 2004-10-01, identification on X samples after that (rows), for sample lengths of Y (columns). Identification with 5-gram models. Each cell shows how many people out of 63 we identified right, given X samples of length Y each past the training date. We can identify majority of the people reliably, and identification quality stays robust far into the future.

ability mass to missing n-grams. Since there are not enough contexts for Kneser-Ney smoothing, which is best for English, we use Witten-Bell smoothing instead. The original MIT Reality data is provided as a MySQL dump. We extract cell tower IDs from the cellspan table and create a separate text file for each person, containing only the IDs in the sequence observed for that person. That file is fed to SRILM and an n-gram model is created for each person, out of N_P= 89 people in total. Then we start N_P TCP person-servers, one for each person-model, to answer queries on perplexity of the incoming phrases in this person's model. We interface SRILM's C++ code to our OCaml-based system, which allows for high-level modelling at high speed, in parallel (at process level). The code is available at [11]. We chose 5-gram order for our models, following the well-established result from natural language processing. OCaml [12] is a high-level functional language which is used for many natural language applications such as creating a Sanskrit dictionary and text segmentation system [7]. It allows for an extensible and compact representation of our methods at native C++ speed.

A. Ranking

We split the data into training and test sets as past and future by a cutoff date, and take samples for identification from the future, keeping the originating person ID for evaluating identification precision, but unknown during identification process itself. When a sample comes, we feed it to each person-server and get back the perplexity, recording the resulting tuple as *(person, perplexity)*. Once all tuples are returned by the servers, we sort them in the order of ascending perplexity, and establish the position of the original person. We call this the self-rank, or just rank. If identification is correct, the original person will come on top with lowest perplexity, having the rank 0. If the rank is more than 0, it means that for the given sample, there are one or more people who are "more like you than yourself," which can be used for clustering.

B. N-gram Identification Results

We pick a date as a cutoff, in our case 2004-10-01, allowing between one and two months of data before that and between five and six months after, and the train our language models on the "past." For each person, we pick 1,000 samples of a fixed length from the "future," for a set of lengths, and run the n-gram identification algorithm. The results of the identification are shown in Table III.

We very quickly reach the top precision achievable with n-gram models, 54/63. A few runs are usually needed to increase the precision to its asymptotic ceiling for a given sample length, but for the longer sample lengths, 50 or 100, one or two samples may be enough. Interestingly, there are always some folks who are hard to identify, at any number of samples or sample sizes. It would be interesting to know why they are hard to model, and if other methods could identify them better.

C. Prediction with N-gram Models

N-gram models can be used for text generation. When generating the next word, the vocabulary is sampled so that it corresponds to the expected n-gram probabilities given the previously observed context. We have contributed to the SRILM generation routine so that it can take an existing context and generate a given-length sentence. Then we conducted an experiment, predicting a person's possible futures and comparing them with what actually transpired. We took person number 38 and considered samples of length 10, and took the first half to predict the second half and compare. We generate a 1,000 possible futures for each prefix, and compare each state (position) of a predicted suffix to the actual one, according to the following metrics.

- **pos.choices** – positional choices: How many options do we have at each step? Count the number of different states predicted at each position across all the simulated suffixes
- **pos.hits** – positional hits: How many times do we get a position right? Count how many simulated suffixes coincide with the actual one at each position
- **sum.hits** – sum of hits: How many total hits do we have per each predicted suffix? This can range from 0 to the length of the suffix, 5 in our case

An example of this process: given the sample *35 1 35 38 3 — 35 3 35 1 35*, with the separator dividing the prefix and the (actual) suffix, we might generate a predicted suffix *35 3 63 38 35*, with $sum.hits = 3$ here. The statistics above are summary for all 1,000 generated suffixes, and are shown in Table IV.

These results can be evaluated in comparison to the fertility of different order n-grams for the person at hand. Fertility of n-grams of orders 1 to 5 are shown in Table V. We see that *pos.choices* is much less, at each position, than unigram fertility. The first "next" position is predicted correctly 64% of the time. Although 19% of all predicted suffixes have no pos.hits at all, the other 81% have some – with more than 53% having 1 or 2, 18% getting 3 out of 5 positions correctly, and 9% getting 4 out of 5.

TABLE IV
PREDICTION OVER 1,000 SIMULATED SUFFIXES

	0	*1*	*2*	*3*	*4*	*5*
pos.choices		27	50	72	78	83
pos.hits		644	373	326	96	284
sum.hits	190	279	252	181	93	5

Predicting the sample *35 1 35 38 3 — 35 3 35 1 35*, subject 38. We hit the first position with about 0.64 accuracy, and get 2 out of 5 hits per suffix quarter of the time.

TABLE V
PERSON CORPUS FERTILITY FOR n-GRAMS OF ORDER 1 TO 5 (SUBJECT 38)

n-gram order	*# n-grams of that order*
1	854/32630
2	2381
3	1716
4	2322
5	2530

The unigram fertility shows actual at 854, and adjusted at 32630 – the latter is the unigram fertility of all people-corpora combined, as required for perplexities to be comparable. People are highly constrained, and lack of n-gram growth shows structure in their movement analogous to that of the natural languages, even in the absence of the "sentence breaks."

VII. IDENTIFICATION WITH SUFFIX TREES

A suffix tree is a compressed trie of all suffixes of a given string [19]. A generalized suffix tree (GST) is a suffix tree of a set of N strings, concatenated together with unique and distinct separators in between. It provides a compact representation of a text corpus, storing N strings together with common prefixes shared, and unique suffixes identifying the original strings containing them. We use an implementation of the GST which stores the set of the IDs of the strings "passing" through a node, and the ID of the string ending in a leaf. Using it, we developed a method of author identification by noticing and counting person-unique subsequences in the sample.

A. Suffix Identification Algorithm

For each subsequence in the sample, we track it to either a leaf or a node of the GST. The node contains IDs of the original strings (people-corpora) which went through it, and we add those to Node counts in favor of those people possibly generating the sample. Each leaf contains an ID of a single potential author, whose Leaf count is then increased. When we're done, we have Leaf and Node counts for each potential author, and take the top Leaf one with a top Node breaking the tie, if available. We get several samples from each person for evaluation and pick the author by a majority vote on all samples by the above procedure.

B. Suffix System Setup

We use a very fast OCaml suffix tree implementation by Sébastien Ferré [3], generalized by us to random alphabets and available as free software [4]. It uses an integer set storing ranges compactly, and the whole of MIT Reality training data (up to the cutoff) fit into about 350 MB of RAM. The natively

TABLE VI
SUFFIX TREE-BASED IDENTIFICATION OF MIT REALITY DATA

batch size \ *sample length*	*10*	*20*	*50*	*100*
1	7	20	19	26
2	12	21	22	25
3	15	25	23	27
5	19	30	31	37
10	23	33	39	42
20	33	35	45	51
50	39	43	50	53
100	45	47	54	53
1000	48	48	52	52

Training on 63 subjects up to 2004-10-01, identification on X samples after that (rows), for sample lengths of Y (columns). Each cell shows how many people out of 63 we identified right, given X samples of length Y each past the training date. While lagging behind n-gram identification a bit, suffix tree quickly catches up with either the sequence length growth or the batch size, and tops off at the precision similar to n-grams. It indicates a ceiling specific to the data itself.

ocamlopt-compiled executable builds the tree in less than a minute, with the longest runs on the full 1,000-sample batches taking less than 4 minutes on a MacBook Pro with the Intel Core 2 Duo 2.4 GHz processor and 4 GB RAM, running sequentially on a single core. In comparison, the full run of the n-gram system on all 63 batches of 1,000 samples each, for the samples of length 100, takes about 6 hours on the same machine, using both cores – but it also selects all of the samples to be used by the subsequent suffix tree identification.

C. Suffix Identification Results

The results are shown in Table VI. The suffix tree is very fast to build and identifies a sample directly, instead of perplexity comparison with all other candidates as in the n-gram model. However, since it counts unique subsequences, it may underperform for those people who share lots of subsequences, especially at shorter sample sizes. When we increase either the sample size or the batch size, we quickly reach the performance of the n-gram model. Suffix trees also represent a whole compressed set of corpora at once, thus allowing for comparisons of the corpora at different points in time, e.g. with the metrics from [16].

VIII. CONCLUSIONS

We presented a life language framework for human behavior modelling, and two algorithms for identification with it. Both language model and suffix tree-based identification show high precision with enough samples, implying that the underlying patterns are separable and sufficiently unique to individuals.

N-gram models achieve high precision very quickly, at 1-2 samples, while suffix trees need more than 10 samples to get to the high precision, but work faster and also identify author-unique subsequences. Prediction with n-grams shows reasonable areas for the future locations, and generally provides much narrower corridors for possible trajectories than generally available, hinting there's a lot of predictability out there waiting to be mined.

We also found that a natively compiled OCaml identification system is comparable to C++ in speed and efficiency, and interoperates well with SRILM – an established C++ library for n-gram modelling, while allowing for easier adaptation to different classes of data. We provide our functional framework for sensor data identification as free software at [4], [11] and hope it can be useful in author-sequence identification in other domains.

IX. FUTURE WORK

We plan to extend the language of life analogy further, to leverage more of the NLP methods. We can treat IDs as letters, not words, and identify words as frequent item sets. Further, we can try shallow parsing, to determine the relationships between segments. Fertility analysis can help gauge when we have enough sensor data to have a representative corpus, and also distinguish jumps – injection of new IDs – as natural segment separators. Suffix trees can be treated probabilistically, allowing for their own prediction techniques. We'll further flesh out the HUMBLE framework, identifying change in behavior analogously to Kullback-Leibler distance between the language models built over windows in time, sliding over corpora. The ultimate goal, of course, is understanding the language of life in the same way as computational linguistics approaches natural language.

X. ACKNOWLEDGMENTS

This work results from a research program in the Institute for Security, Technology, and Society at Dartmouth College, supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, Metro-Sense; Scalable Secure Sensor Systems. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security.

We'd like to thank Dmitri Levonian, the founder of Corpus Technologies, for starting the fertility studies as a means to measure when collecting "enough" of a corpus is really – or already – enough. Andreas Stolcke, the author and a decade-long maintainer of SRILM, helped with the APIs and integrated our improvements. We thank Vince Berk and the HUMBLE group at Thayer for casting and developing these ideas in the (fun) context of quantitative human behavior modeling.

REFERENCES

[1] S. Chen and J. Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, August 1998.

[2] N. Eagle and A. Pentland. MIT Reality data. http://reality.media.mit.edu/, 2005.

[3] S. Ferré. An efficient generalized suffix tree implemenattion in OCaml. http://www.irisa.fr/LIS/ferre/software.en.html.

[4] S. Ferré and A. Khrabrov. Generalized suffix tree in OCaml over generic alphabets, Free Software. http://github.com/alexy/suffix/tree/master/.

[5] M. C. González, C. A. Hidalgo, and A.-L. Barabási. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, Jun 2008.

[6] J. Goodman. A bit of progress in language modeling, extended version. Technical Report MSR-TR-2001-72, Microsoft Research, 2001.

[7] G. Huét. INRIA Sanskrit portal. http://sanskrit.inria.fr/.

[8] H. Huo and V. Stojkovic. A partition-based suffix tree construction and its applications. In W. Bednorz, editor, *Advances in Greedy Algorithms*, page 586. November 2008.

[9] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1997.

[10] D. Khmelev. Text authorship recognition using Markov chains. *Vestnik of the Moscow State University*, 9(2):115–126, 2000. Also available as http://www.philol.msu.ru/~lex/khmelev/published/vestnik/vestnik2000.pdf.

[11] A. Khrabrov. Life language identification code, Free Software. http://github.com/alexy/life-language/tree/master/.

[12] X. Leroy. The OCaml programming language. http://caml.inria.fr/.

[13] D. Levonian, A. Khrabrov, and S. Rubakov. N-gram fertility of a natural language corpus. Technical report, Corpus Technologies, 2008. Published as http://suffix.com/tech-reports/fertility/.

[14] C. D. Manning and H. Shütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[15] A. Markov. An example of a statistical study on the text of "Eugene Onegin", illustrating connecting observations into a chain. *Annals of the Russian Imperial Academy of Science*, X(3), 1913.

[16] K. Rieck, P. Laskov, and S. Sonnenburg. Computation of similarity measures for sequential data using generalized suffix trees. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 1177–1184. MIT Press, Cambridge, MA, 2007.

[17] N. F. Sandell, R. Savell, D. Twardowski, and G. Cybenko. HBML: A language for quantitative behavioral modeling in the human terrain. In *2nd International Workshop on Social Computing, Behavioral Modeling, and Prediction in Phoenix, AZ*. 2009.

[18] A. Stolcke. SRILM – an extensible language modeling toolkit. In *Proc. Intl. Conf. Spoken Language Processing, Denver, Colorado*. September 2002. Code is available at http://www.speech.sri.com/projects/srilm/.

[19] P. Weiner. Linear pattern matching algorithm. In *14th Annual IEEE Symposium on Switching and Automata Theory*, pages 1–11. 1973.

[20] I. Witten and T.C.Bell. The zero-frequency problem: Estimating probability of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, 1991.