Realization of Common Statistical Methods in Computational Linguistics with Functional Automata

Stefan Gerdjikov
Faculty of Mathematics and Informatics,
Sofia University
5 James Bourchier blvd.,
1164 Sofia, Bulgaria
st_gerdjikov@abv.bg

Petar Mitankin
Faculty of Mathematics and Informatics,
Sofia University
5 James Bourchier blvd.,
1164 Sofia, Bulgaria
pmitankin@fmi.uni-sofia.bg

Vladislav Nenchev
Faculty of Mathematics and Informatics,
Sofia University
5 James Bourchier blvd.,
1164 Sofia, Bulgaria
lucifer.dev.0@gmail.com

Abstract

In this paper we present the *functional automata* as a general framework for representation, training and exploring of various statistical models as LLM's, HMM's, CRF's, etc.

Our contribution is a new construction that allows the representation of the derivatives of a function given by a functional automaton. It preserves the natural representation of the functions and the standard product and sum operations of real numbers. In the same time it requires no additional overhead for the standard dynamic programming techniques that yield the computation of a functional value.

1 Introduction

Statistical models such as n-gram language models (Chen and Goodman, 1996), hidden Markov models (Rabiner, 1989), conditional random fields (Lafferty et al., 2001), log-linear models (Darroch and Ratcliff, 1972) are widely applied in the natural language processing in order to approach various problems, e.g. parsing (Sha and Pereira, 2003), speech recognition (Juang and Rabiner, 1991), statistical machine translation (Brown et al., 1993). Different statistical models perform differently on different tasks. Thus in order to find the best practical solution one might need to try several approaches before getting the desired effect. Disposing on a general framework that allows the flexibility to change the statistical model or/and training scheme would spend much efforts and time.

Focusing on this pragmatical problem, we propose the *functional automata* as a possible solution. The basic idea is to consider the mathematical expressions of sums and products arising in the statistical models as regular expressions. Thus regarding the functions in these expressions as individual characters, the sums as unions and the products as concatenation, we get the desired correspondence. The relation between a particular statistical model and a functional automaton for its representation is then rather straightforward.

The training of the statistical models is in a way more involved. Most of the approaches require a gradient method that estimates the best model parameters. To this end one needs to have an efficient representation not only of the function used by the model but also of its (partial) derivatives.

To solve similar problem Eisner and Li introduce first-order and second-order expectation semirings. In (Jason Eisner, 2002; Zhifei Li and Jason Eisner, 2009) it is shown how derivatives of functions arising in statistical models can be represented. This is achieved by the means of an algebraic construction that: (i) considers pairs of functions (first-order expectation semiring) and quadruples of functions (second-order expectation semiring); (ii) introduces an operation on pairs and quadruples, respectively, of functions that replaces the multiplication and is used to simulate the multiplication of first- and second-order derivatives, respectively. Thus the higher the order of the derivatives in interest, the more complex would be the necessary expectation semiring and the operations that it would require.

In the current paper we propose an alternative approach. It is based on a combinatorial construction that allows preserving both: (i) manipulation with single functions and (ii) the usage of the standard multiplication and addition of real numbers. Thus we get a uniform representation of functions, their first- and higher order derivatives. Our approach requires the same storage as the approach in (Jason Eisner, 2002; Zhifei Li and Jason Eisner, 2009) and enables the same efficiency for the traversal procedures described in (Zhifei Li and Jason Eisner, 2009).

In Section 3 we show that the values of a function represented by an acyclic functional automaton can be efficiently computed by the means of a standard dynamic programming technique. We further describe how to construct functional automata for the partial derivatives of F by given functional automaton representing F. We show in Sections 2 and 6 that such automata can be used for training log-linear models, hidden Markov models and conditional random fields. We only require that the objective function is represented via functional automata. In Section 5 we present a construction of functional automaton for a log-linear model where one of the feature functions uses an n-gram language model (Chen and Goodman, 1996).

In Section 7 we present evaluation of a developed system, based on functional automata, on the tasks of (i) noisy historical text normalization and (ii) OCR postcorrection.

2 Log-linear models

We consider the task of automatic normalization of Early Modern English texts. In the next two paragraphs we define some notions related to this task. We use them afterwards to formulate typical problems of training and search that can be effectively solved by functional automata.

Given a *source* text s, say $s =$ *theldest sonn hath bin kild*, and the goal is to find the most relevant modern English equivalent of s. A *candidate generator* is an algorithm that for a fixed source word or sequence of words, say $s_i s_{i+1} \dots s_{i+k}$, generates finite number of *normalization candidates* and supplies each normalization candidate, c, with a conditional probability, $p_{cg}(c \mid s_i s_{i+1} \dots s_{i+k})$. Hence we can assume that the candidate generator provides the information in the form of Table 1. In this sense the candidate generator corresponds to the word-to-word or phrase-to-phrase translation tables in statistical machine translation systems (Koehn et al., 2003). From the candidates we construct possible normalization *targets*: *eldest sun hat been kid, the eldest soon has bean killed, the eldest son has been killed* etc. For normalization of texts produced by OCR system from noisy

source word	set of target candidates
$theldest$	$\{\langle the\ eldest, 0.75\rangle, \langle eldest, 0.25\rangle\}$
$sonn$	$\{\langle son, 0.92593\rangle, \langle soon, 0.03704\rangle, \langle sun, 0.03704\rangle\}$
$hath$	$\{\langle hat, 0.0088\rangle, \langle hats, 0.0044\rangle, \langle has, 0.9868\rangle\}$
bin	$\{\langle bin, 0.1\rangle, \langle been, 0.8\rangle, \langle bean, 0.1\rangle\}$
$kild$	$\{\langle kid, 0.01\rangle, \langle killed, 0.99\rangle\}$

Table 1: Source words and their corresponding set of candidates provided by the candidate generator. Each target candidate c for the source word s_i is associated with a probability $p_{cg}(c \mid s_i)$.

historical documents the candidate generator could take into account both typical OCR errors and historical spelling variations, (Reffle, 2011) or can use directly automatically extracted spelling variations, for example (Gerdjikov et al., 2013).

A *normalization pair* is a pair $p = \langle w, c\rangle$ such that the sequence of target words c is a normalization candidate for the sequence of source words w. We call w *left side* and c *right side* of the normalization pair p. The left and the right sides of p are denoted $l(p)$ and $r(p)$ respectively. In our example some of the normalization pairs are $\langle theldest, eldest\rangle$, $\langle theldest, theeldest\rangle$, $\langle kild, killed\rangle$, etc. A *normalization alignment* from s to t, denoted $s \rightarrow t$, is a sequence of normalization pairs $p_1 p_2 \dots p_k$ such that $s = l(p_1)l(p_2)\dots l(p_k)$ and $t = r(p_1)r(p_2)\dots r(p_k)$. The i-th normalization pair p_i of the alignment $s \rightarrow t$ is denoted $(s \rightarrow t)_i$. The length k of the alignment is denoted $|s \rightarrow t|$. Thus a possible normalization alignment in our example, from $s =$ *theldest sonn hath bin kild* to $t =$ *eldest sun hat been kid* is ⟨*theldest*, *eldest*⟩ ⟨*sonn*, *sun*⟩⟨*hath*, *hat*⟩⟨*bin*, *been*⟩⟨*kild*, *kid*⟩. We denote with A_s the set of all normalization alignments from s. Note that A_s is always finite, because the number of normalization candidates for each sequence $s_i s_{i+1} \dots s_{i+k}$ of source words is finite.

Problem. Given a training corpus of normalization alignments *train* a log-linear model that combines the candidate generator with an n-gram statistical language model. Once the model is trained, *find* a best normalization alignment $s \rightarrow t$ for a given source s.

Firstly, we consider the case where $n = 1$, i.e. we have a monogram language model which assigns a nonzero probability $p_{lm}(t_i)$ to each target word t_i. The general case of arbitrary n-gram language model is postponed to Section 5. There are two feature functions: $h_{lm}(s \to t) = \log \prod_{i=1}^{|t|} p_{lm}(t_i)$ and $h_{cg}(s \to t) = \log \prod_{i=1}^{|s\to t|} p_{cg}[r((s \to t)_i) \mid l((s \to t)_i)]$. The probability of a normalization alignment $s \to t$ given s is $p_\lambda(s \to t \mid s) =$

$$\frac{\exp[\lambda_{lm} h_{lm}(s \to t) + \lambda_{cg} h_{cg}(s \to t)]}{\sum_{s\to t' \in A_s} \exp[\lambda_{lm} h_{lm}(s \to t') + \lambda_{cg} h_{cg}(s \to t')]},$$

where $\lambda = \langle \lambda_{lm}, \lambda_{cg} \rangle$ are the parameters of the model.

Training. Assume that we have a training corpus T of N normalization alignments, $T = \langle s^{(1)} \to t^{(1)}, s^{(2)} \to t^{(2)}, \ldots, s^{(N)} \to t^{(N)} \rangle$. The training task is to find parameters $\hat{\lambda}$ that optimize the joint probability over the training corpus, $\hat{\lambda} = argmax_\lambda \prod_{n=1}^{N} p_\lambda(s^{(n)} \to t^{(n)} \mid s^{(n)})$.

Search. Once the parameters $\hat{\lambda}$ are fixed, the problem is to find a best normalization alignment $s \to t = argmax_{s\to t' \in A_s}\, p_{\hat{\lambda}}(s \to t')$ for a given input s.

Introducing $e_{s\to t}(\lambda) = \exp[\lambda_{lm} h_{lm}(s \to t) + \lambda_{cg} h_{cg}(s \to t)]$ and

$$Z_s(\lambda) = \sum_{s\to t' \in A_s} e_{s\to t'}(\lambda), \qquad (1)$$

we obtain $\hat{\lambda} = argmax_\lambda\, L(\lambda)$, where

$$L(\lambda) = \sum_{n=1}^{N} [\lambda_{lm} h_{lm}(s^{(n)} \to t^{(n)}) + \lambda_{cg} h_{cg}(s^{(n)} \to t^{(n)}) - \log\, Z_{s^{(n)}}(\lambda)]. \qquad (2)$$

To optimize $L(\lambda)$ we use a gradient method that requires the computation of $L(\lambda)$, $\frac{\partial L}{\partial \lambda_{cg}}(\lambda)$ and $\frac{\partial L}{\partial \lambda_{lm}}(\lambda)$ by given λ. For $i = lm, cg$ we obtain

$$\frac{\partial L}{\partial \lambda_i}(\lambda) = \sum_{n=1}^{N} [h_i(s^{(n)} \to t^{(n)}) - \frac{\frac{\partial Z_{s^{(n)}}}{\partial \lambda_i}(\lambda)}{Z_{s^{(n)}}(\lambda)}]. \qquad (3)$$

One possible choice of first order gradient method for the optimization of L is a variant of the conjugate gradient method that converges to the unique maximum of L for each starting point $\lambda_0 = \langle \lambda_{lm0}, \lambda_{cg0} \rangle$, (Gilbert and Nocedal, 1992).

3 Functional automata

The problem we faced in the previous Section is how to compute $L(\lambda)$ and $\frac{\partial L}{\partial \lambda_i}(\lambda)$ at a given point λ. The computation of the terms $\lambda_i h_i(s^{(n)} \to t^{(n)})$ for $i = cg$ (or $i = lm$) is easy since it requires a single multiplication and $|s^{(n)} \to t^{(n)}|$ (or $|t^{(n)}|$) additions. However the

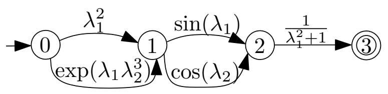


Figure 1: Functional automaton representing the function $F(\lambda_1, \lambda_2) = \lambda_1^2 \sin(\lambda_1)\frac{1}{\lambda_1^2+1} + \lambda_1^2 \cos(\lambda_2)\frac{1}{\lambda_1^2+1} + \exp(\lambda_1\lambda_2^3)\sin(\lambda_1)\frac{1}{\lambda_1^2+1} + \exp(\lambda_1\lambda_2^3)\cos(\lambda_2)\frac{1}{\lambda_1^2+1}$

term $Z_s(\lambda)$ may require much more efforts. It suffices that each source word s_i generates two candidates for the expression in Equation 1 to explode in exponential number of summation terms. Computing the derivatives then becomes even harder. In this Section we present a novel efficient solution to these problems. It is based on a compact representation of the mathematical expressions via *functional automata*.

Imagine, that we have the function $F(\lambda_1, \lambda_2)$ given as an expression: $\lambda_1^2 \sin(\lambda_1)\frac{1}{\lambda_1^2+1} + \lambda_1^2 \cos(\lambda_2)\frac{1}{\lambda_1^2+1} + \exp(\lambda_1\lambda_2^3)\sin(\lambda_1)\frac{1}{\lambda_1^2+1} + \exp(\lambda_1\lambda_2^3)\cos(\lambda_2)\frac{1}{\lambda_1^2+1}$. Let us further assume that we interpret the individual functions λ_1^2, $\cos(\lambda_2)$, $\frac{1}{\lambda_1^2+1}$, etc, as single symbols. If we further interpret the multiplication of functions as concatenation and the addition as union, then the expression for $F(\lambda_1, \lambda_2)$ given above can be viewed as a regular expression for which a finite state automaton can be compiled, see Figure 1. This is the motivation for the following two definitions:

Definition 3.1 Let d be a positive natural number. *Functional automaton* is a quadruple $\mathcal{A} = \langle Q, q_0, \Delta, T \rangle$, where Q is a finite set of states, $q_0 \in Q$ is a start state, Δ is a finite multiset of transitions of the form $q \xrightarrow{W} p$ where $p, q \in Q$ are states and $W : \mathbb{R}^d \to \mathbb{R}$ is a function and $T \subseteq Q$ is a set of final states.

Definition 3.2 Let $\mathcal{A} = \langle Q, q_0, \Delta, T \rangle$ be an acyclic functional automaton (AFA). A *path π from p_0 to p_k in $\mathcal{A}$* is a sequence of $k \geq 0$ transitions $\pi = p_0 \xrightarrow{W_1} p_1 \xrightarrow{W_2} p_2 \ldots p_{k-1} \xrightarrow{W_k} p_k$. *The label of* π is defined as $l_\pi = \prod_{j=1}^{k} W_j$. If π is empty ($k = 0$), then $l_\pi = 1$. A *successful path* is a path from q_0 to a final state $q \in T$. *The function $F_\mathcal{A} : \mathbb{R}^d \to \mathbb{R}$ represented by $\mathcal{A}$* is defined as $F_\mathcal{A} = \sum_{\pi \text{ is a successful path in } \mathcal{A}} l_\pi$.

Since $\mathcal{A}$ is acyclic, the number of successful paths is finite and $F_\mathcal{A}$ is well defined.

target word	*the*	*eldest*	*son*	*soon*	*sun*
probability	0.017	0.00002	0.0003	0.0005	0.0002

target word	*hat*	*hats*	*has*	*bin*
probability	0.0001	0.00002	0.002	0.000005

target word	*been*	*bean*	*kid*	*killed*
probability	0.003	0.000005	0.00002	0.0001

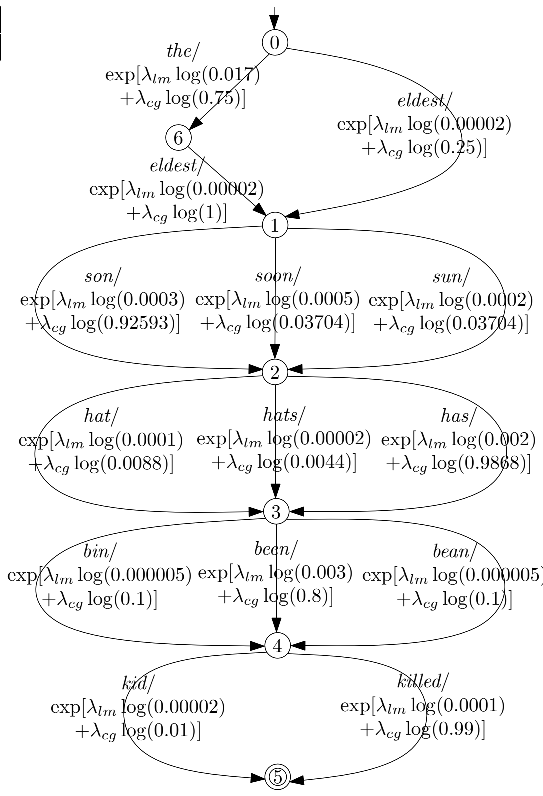
Table 2: Target words and their language model probabilities.

Classical constructions for *union* and *concatenation* of automata (Hopcroft and Ullman, 1979) can be adapted for functional automata. If $\mathcal{A}$ is the result of the union (concatenation) of $\mathcal{A}_1$ and $\mathcal{A}_2$, then $F_{\mathcal{A}} = F_{\mathcal{A}_1} + F_{\mathcal{A}_2}$ ($F_{\mathcal{A}} = F_{\mathcal{A}_1} \cdot F_{\mathcal{A}_2}$).

3.1 Computation of a function $F_{\mathcal{A}}$ represented by an AFA $\mathcal{A}$

In order to efficiently compute $F_{\mathcal{A}}(\lambda)$ for a given $\lambda = \langle \lambda_1, \lambda_2, \ldots, \lambda_n \rangle$, we use standard dynamic programming. Without loss of generality we assume that $\mathcal{A} = \langle Q, q_0, \Delta, T \rangle$ has only one final state and each transition in A belongs to some successful path. Firstly, we sort topologically the states of the automaton $\mathcal{A}$ in decreasing order. Let $p_1, p_2, \ldots, p_{|Q|}$ be one such order of the states, i.e. (i) $p_1 \in T$ is the only one final state, (ii) $p_{|Q|} = q_0$ is the start state and (iii) if there is a transition from p_i to p_j then $j < i$. For example for the automaton on Figure 1 we obtain $3, 2, 1, 0$. Afterwards for each state p_j we compute a value v_j in the following way: $v_1 = 1$ and $v_{j+1} = \sum_{p_{j+1} \xrightarrow{W(\lambda)} p_k} W(\lambda) \cdot v_k$. Eventually $F_{\mathcal{A}}(\lambda) = v_{|Q|}$. If the computation of $W(\lambda)$ by given λ takes time $O(1)$ for all label functions W, then the time for the computation of $F_{\mathcal{A}}(\lambda)$ is $O(|\Delta|)$.

Now we focus on the problem how to compute $Z_s(\lambda)$ at a given point λ, see Equation 1. We illustrate how $Z_s(\lambda)$ can be represented by an AFA, $\mathcal{A}_s$, on the example from Section 2, $s =$ *theldest sonn hath bin kild*. Table 1 lists the sets of candidates in modern English for each source word s_i. Table 2 presents the language model probabilities for each target word. Given this data we represent the possible normalization alignments via an acyclic two-tape automaton, see Figure 2. This automaton can be considered as a string-to-weight transducer (Mohri, 1997) parameterized with λ_{lm} and λ_{cg}. Specifically, each path from state $i-1$ to state i, $1 \leq i \leq |s|$, corresponds to a target candi-



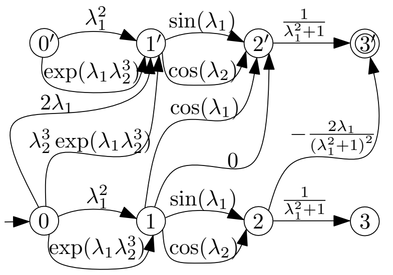
0
the/ $\exp[\lambda_{lm}\log(0.017) + \lambda_{cg}\log(0.75)]$
eldest/ $\exp[\lambda_{lm}\log(0.00002) + \lambda_{cg}\log(0.25)]$
6
eldest/ $\exp[\lambda_{lm}\log(0.00002) + \lambda_{cg}\log(1)]$
1
son/ $\exp[\lambda_{lm}\log(0.0003) + \lambda_{cg}\log(0.92593)]$
soon/ $\exp[\lambda_{lm}\log(0.0005) + \lambda_{cg}\log(0.03704)]$
sun/ $\exp[\lambda_{lm}\log(0.0002) + \lambda_{cg}\log(0.03704)]$
2
hat/ $\exp[\lambda_{lm}\log(0.0001) + \lambda_{cg}\log(0.0088)]$
hats/ $\exp[\lambda_{lm}\log(0.00002) + \lambda_{cg}\log(0.0044)]$
has/ $\exp[\lambda_{lm}\log(0.002) + \lambda_{cg}\log(0.9868)]$
3
bin/ $\exp[\lambda_{lm}\log(0.000005) + \lambda_{cg}\log(0.1)]$
been/ $\exp[\lambda_{lm}\log(0.003) + \lambda_{cg}\log(0.8)]$
bean/ $\exp[\lambda_{lm}\log(0.000005) + \lambda_{cg}\log(0.1)]$
4
kid/ $\exp[\lambda_{lm}\log(0.00002) + \lambda_{cg}\log(0.01)]$
killed/ $\exp[\lambda_{lm}\log(0.0001) + \lambda_{cg}\log(0.99)]$
5

Figure 2: The functional automaton $\mathcal{A}_{theldest\ sonn\ hath\ bin\ kild}$ is obtained by removing the words from the transition labels.

date c for the i-th source word s_i and has a label $exp[\lambda_{cg} log(p_{cg}(c \mid s_i)) + \lambda_{lm} log(p_{lm}(c))]$. On our example, for $i \geq 2$ each such path consists of a single transition, because the candidates are single words. In order to represent the candidate $the\ \ eldest$ we use the additional state 6. The transition from 0 to 6 corresponds to the first word the of the candidate and accumulates the whole probability $p_{cg}(the\ \ eldest \mid theldest) = 0.75$. The transition from 6 to 1 corresponds to the second word $eldest$ of the candidate. It should be clear that removing the target words from the transitions, we obtain the AFA $\mathcal{A}_s$ representing $Z_s(\lambda)$. For each alignment $s^{(n)} \to t^{(n)}$ from the training corpus we build a separate functional automaton, like the one on Figure 2, representing $Z_{s^{(n)}}(\lambda)$. Thus we have N automata that we use to compute $L(\lambda)$ via Equation (2).

3.2 Computation of partial derivates via AFA

Our next goal is to compute the partial derivates $\frac{\partial L}{\partial \lambda_i}(\lambda)$. Let us turn back to the function $F(\lambda_1, \lambda_2)$ represented by the automaton on Figure 1. We show how to construct a functional automaton for



λ_1^2
$\sin(\lambda_1)$
$\frac{1}{\lambda_1^2+1}$
$0'$
$1'$
$2'$
$3'$
$\exp(\lambda_1\lambda_2^3)$
$\cos(\lambda_2)$
$2\lambda_1$
$\cos(\lambda_1)$
$\lambda_2^3\exp(\lambda_1\lambda_2^3)$
$-\frac{2\lambda_1}{(\lambda_1^2+1)^2}$
0
λ_1^2
$\sin(\lambda_1)$
$\frac{1}{\lambda_1^2+1}$
0
1
2
3
$\exp(\lambda_1\lambda_2^3)$
$\cos(\lambda_2)$

Figure 3: A functional automaton for the partial derivative of $F(\lambda_1, \lambda_2)$.

$\frac{\partial F}{\partial \lambda_1}(\lambda_1, \lambda_2)$. Let $G(\lambda_1, \lambda_2) = \lambda_1^2 \sin(\lambda_1)\frac{1}{\lambda_1^2+1}$ be the first of the four summation terms of F. The partial derivative $\frac{\partial G}{\partial \lambda_1}$ can be written as a sum of three terms: $\frac{\partial(\lambda_1^2)}{\partial \lambda_1}\sin(\lambda_1)\frac{1}{\lambda_1^2+1}$, $\lambda_1^2\frac{\partial(\sin(\lambda_1))}{\partial \lambda_1}\frac{1}{\lambda_1^2+1}$ and $\lambda_1^2\sin(\lambda_1)\frac{\partial(\frac{1}{\lambda_1^2+1})}{\partial \lambda_1}$. Each of the summation terms differs from the original expression for $G(\lambda_1, \lambda_2)$ in exactly one multiplier whose partial derivative with respect to λ_1 is computed. Thus in order to construct a functional automaton for $\frac{\partial F}{\partial \lambda_1}$ we can take two disjoint copies of the original functional automaton, see Figure 3, and set transitions between them in order to reflect the partial derivatives with respect to λ_1 of the single multipliers. The general result is presented in the following Proposition:

Proposition 3.3 *Let $\mathcal{A}$ be an AFA with k states and t transitions and let $\mathcal{A}' = \langle Q', q_0', \Delta', T'\rangle$ be a disjoint copy of $\mathcal{A}$. If the partial derivatives $\frac{\partial W}{\partial \lambda_i}$ exist for each transition $q \overset{W(\lambda_1,\lambda_2,\ldots,\lambda_d)}{\longrightarrow} p$ in $\mathcal{A}$, then $\mathcal{B} = \langle Q \cup Q', q_0, \Delta \cup \Delta' \cup \{q \overset{\frac{\partial W}{\partial \lambda_i}}{\to} p' \mid q \overset{W}{\to} p \in \Delta\}, T'\rangle$ is an AFA with $2k$ states, $3t$ transitions and $F_{\mathcal{B}} = \frac{\partial F_{\mathcal{A}}}{\partial \lambda_i}$.*

Sketch of proof. We have $\frac{\partial F_{\mathcal{A}}}{\partial \lambda_i} = \sum_{\pi \text{ is a successful path in } \mathcal{A}} \frac{\partial l_\pi}{\partial \lambda_i} = \sum_{\substack{\pi = q_0 \xrightarrow{W_1} q_1 \ldots q_{m-1} \xrightarrow{W_m} q_m \\ \text{is a successful path in } \mathcal{A}}} \sum_j \pi_{(j,i)}$, where $\pi_{(j,i)} = W_1 \ldots W_{j-1}\frac{\partial W_j}{\partial \lambda_i}W_{j+1}\ldots W_m$. There is a one-to-one correspondence between the successful paths in $\mathcal{B}$ and the terms $\pi_{(j,i)}$ in the above summation. $\square$

Let us note that the construction presented in Proposition 3.3 can be iterated i times in order to build a functional automaton with $2^i k$ states and $3^i t$ transitions for each i-th order partial derivate of $F_{\mathcal{A}}$. Thus we can build functional automata with $4k$ states and $9t$ transitions for $\frac{\partial^2 F_{\mathcal{A}}}{\partial \lambda_i \lambda_j}$. This gives the possibility to use some second order gradient method in the training procedure. Note that if the computation of $W(\lambda)$ for a given λ and all label functions, W, takes constant time, then using functional automata we achieve an $O(t)$-time computation of both $\frac{\partial F_{\mathcal{A}}}{\partial \lambda_i}(\lambda)$ and $\frac{\partial^2 F_{\mathcal{A}}}{\partial \lambda_i \lambda_j}(\lambda)$.

4 Search procedure

By given source sequence s we want to find best alignment $s \to t = argmax_{s \to t' \in A_s}\, p_{\hat{\lambda}}(s \to t') = argmax_{s \to t' \in A_s}\ e_{s \to t'}(\hat{\lambda})$. For this purpose we use again a standard dynamic programming procedure on the automaton $\mathcal{A}_s$ representing the function $Z_s(\lambda)$, Figure 2. The only difference with the procedure described in Subsection 3.1 is that instead of summation over all transtions from the current state we need to take maximum and to mark a transition that gives the maximum. Finally the successful path of marked transitions represents a best alignment. Actually this procedure corresponds to the backward version of the Viterbi decoding algorithm (Omura, 1967). If the computation of $W(\lambda)$ by given λ takes time $O(1)$ for all label functions W, then the search procedure is linear in the number of the transitions in the functional automaton.

5 n-gram language models

In this Section we generalize the constructions of the automaton $\mathcal{A}_s$ from Section 3 and 4 to the case of an arbitrary n-gram language model, $n > 1$. In this case $h_{lm}(s \to t) = \log \prod_{i=1}^{|t|} p_{lm}(t_i \mid t_{i-n+1}t_{i-n+2}\ldots t_{i-1})$. We construct an automaton representing $Z_s(\lambda)$ as follows. Firstly, we build automaton $\mathcal{A}_1$ that represents the function $Z_s(\langle 0, \lambda_{cg}\rangle) = \sum_{s \to t' \in A_s} \exp[\lambda_{cg}h_{cg}(s \to t')]$. Each transition in $\mathcal{A}_1$ is associated with a target word, see Figure 2. Now we would like to add $\exp[\lambda_{lm}\log(p_{lm}(t_i \mid t_{i-n+1}t_{i-n+2}\ldots t_{i-1}))]$ to the label of each transition associated with t_i. However the problem is that there may be multiple sequences of preceding words $t_{i-n+1}t_{i-n+2}\ldots t_{i-1}$ for one and the same transition. For example for $n = 3$ on Figure 2 for the transition associated with $t_i = has$ from state 2 to state 3 there are three different possible pairs of preceding words $t_{i-2}t_{i-1}$: *eldest son*, *eldest* *soon* and *eldest sun*. We overcome this problem of ambiguity by extending $\mathcal{A}_1 = \langle Q_1, q_1, \Delta_1, T_1\rangle$ to equivalent automaton $\mathcal{A}_2$ in which for each state the sequence of $n-1$ preceding words is uniquely determined. The set of states of $\mathcal{A}_2$ is $Q_2 = \{\langle w_1 w_2 \ldots w_{n-1}, q\rangle \mid q \in Q_1$ and $w_1 w_2 \ldots w_{n-1}$ is a sequence of preceding words for q in $\mathcal{A}_1\}$. The set of transitions of $\mathcal{A}_2$ is $\Delta_2 = \{\langle w_1 w_2 \ldots w_{n-1}, q'\rangle \xrightarrow{W} \langle w_2 \ldots w_{n-1} w_n, q''\rangle \mid$ transition $q' \xrightarrow{W} q'' \in \Delta_1$ is associated with $w_n\}$. In $\mathcal{A}_2$ the transition $\langle w_1 w_2 \ldots w_{n-1}, q'\rangle \xrightarrow{W} \langle w_2 \ldots w_{n-1} w_n, q''\rangle$ is associated with the word w_n. Finally, from $\mathcal{A}_2$ we construct functional automaton $\mathcal{A}_3$ that represents $Z_s(\langle \lambda_{lm}, \lambda_{cg}\rangle)$ by adding $\exp[\lambda_{lm} \log(p_{lm}(w_n \mid w_1 w_2 \ldots w_{n-1}))]$ to the label of each transition t where w_n is the word associated with t.

If m is an upper bound for the number of correction candidates for every sequence $s_i s_{i+1} \ldots s_{i+k}$, then $|Q_2| \leq m^{n-1}|Q_1|$ and $|\Delta_2| \leq m^{n-1}|\Delta_1|$.

6 Other statistical models

In this section we apply the technique developed in Sections 3 and 4 to other statistical models.

Conditional random fields. A linear-chain CRF serves to assign a label y_i to each the observation x_i of a given observation sequence x. We assume that the observations x_i belong to a set X and the labels y_i belong to a finite set Y. We shall further consider that the probability measure of a linear-chain CRF with $|x|$ states is

$$p_\lambda(y \mid x) = \frac{\exp[\sum_{i=2}^{|x|}\sum_{j=1}^{K} \alpha_j f_j(y_{i-1}, y_i, x, i) + \sum_{i=1}^{|x|}\sum_{j=1}^{K} \beta_j g_j(y_i, x, i)]}{Z_x(\lambda)}$$

where $|x| = |y|$, $f_j : Y \times Y \times X^* \times \mathbb{N} \to \mathbb{R}$ and $g_j : X^* \times \mathbb{N} \to \mathbb{R}$ are predefined feature functions, $\lambda = \langle \alpha_1, \alpha_2, \ldots, \alpha_K, \beta_1, \beta_2, \ldots, \beta_K\rangle$ are parameters and $Z_x(\lambda) = \sum_{y \in Y^{|x|}} \exp[\sum_{i=2}^{|x|}\sum_{j=1}^{K} \alpha_j f_j(y_{i-1}, y_i, x, i) + \sum_{i=1}^{|x|}\sum_{j=1}^{K} \beta_j g_j(y_i, x, i)]$. The training task is similar to the one described in Section 2. We have a training corpus of N pairs $\langle x^{(1)}, y^{(1)}\rangle, \langle x^{(2)}, y^{(2)}\rangle, \ldots, \langle x^{(N)}, y^{(N)}\rangle$ and we need to find the parameters $\hat{\lambda} = argmax_\lambda \prod_{n=1}^{N} p_\lambda(y^{(n)} \mid x^{(n)})$. Formulae very similar to (2) and (3) can be derived. Thus the main problem is again in the computation of the term $Z_x(\lambda)$. In (Lafferty et al., 2001) $Z_x(\lambda)$ is represented as an entity of a special matrix which is obtained as a product of $|x|+1$ matrices of size $(|Y|+2) \times (|Y|+2)$. The states of an AFA $\mathcal{A}_x$ representing $Z_x(\lambda)$ are as follows: a start state s, a final state f and $|x| \cdot |Y|$ "intermediate" states $q_{i,\gamma}$, $1 \leq i \leq |x|$, $\gamma \in Y$. The transitions are $s \xrightarrow{G} q_{1,\gamma}$ for $G = \exp \sum_{j=1}^{K} \beta_j g_j(\gamma, x, 1)$, $q_{i,\gamma'} \xrightarrow{F} q_{i+1,\gamma''}$ for $F = \exp \sum_{j=1}^{K} [\alpha_j f_j(\gamma', \gamma'', x, i+1) + \beta_j g_j(\gamma'', x, i+1)]$ and $q_{|x|,\gamma} \xrightarrow{1} f$. Transitions with label 0 can be removed from the automaton. If there are many such transitions this could significantly reduce the time for training.

Hidden Markov models. We adapt the notations and the definitions from (Rabiner, 1989). Let $\lambda = \langle A, B, \pi\rangle$ be the parameters of a HMM with R states $S = \{S_1, S_2, \ldots, S_R\}$ and M distinct observation symbols $V = \{v_1, v_2, \ldots, v_M\}$, where $A = \{a_{S_i S_j}\}$ is a $R \times R$ matrix of transition probabilities, $B = \{b_{S_j}(v_k)\}$ are the observation symbol probability distributions and $\pi = \{\pi_{S_j}\}$ is the initial state distribution. The probability of $O_1 O_2 \ldots O_T$ is $p_\lambda(O_1 O_2 \ldots O_T) = \sum_{q_1 q_2 \ldots q_T \in S^T} c(q_1 q_2 \ldots q_T)$, where $c(q_1 q_2 \ldots q_T) = \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) \ldots a_{q_{T-1} q_T} b_{q_T}(O_T)$.

Given a training set of N observations $O^{(1)}, O^{(2)}, \ldots, O^{(N)}$ the optimal parameters $\hat{\lambda} = argmax_\lambda \prod_{n=1}^{N} p_\lambda(O^{(n)})$ have to be determined under the stohastic constraints $\sum_j a_{S_i S_j} = 1$, $\sum_k b_{S_j}(v_k) = 1$ and $\sum_j \pi_{S_j} = 1$. Applying the method of Lagrange multipliers we obtain a new function $F(\lambda, \alpha, \beta, \gamma) = \prod_{n=1}^{N} p_\lambda(O^{(n)}) + \sum_i \alpha_i[(\sum_j a_{S_i S_j}) - 1] + \sum_i \beta_i[(\sum_k b_{S_j}(v_k)) - 1] + \gamma[(\sum_j \pi_{S_j}) - 1]$. For each training observation sequence $O^{(n)}$ with $T^{(n)}$ symbols the function $p_\lambda(O^{(n)})$ can be represented by an AFA $\mathcal{A}_{O^{(n)}}$ with $RT^{(n)} + 2$ states, $R(T^{(n)} + 1)$ transitions and a single final state as follows. We have the start state s, the final state f and $RT^{(n)}$ "intermediate" states q_{t,S_i}, $1 \leq t \leq T^{(n)}$, $1 \leq i \leq R$. The transitions are $s \xrightarrow{\pi_{S_i} b_{S_i}(O_1^{(n)})} q_{1,S_i}$, $q_{t,S_i} \xrightarrow{a_{S_i S_j} b_{S_j}(O_{t+1}^{(n)})} q_{t+1,S_j}$ and $q_{T^{(n)},S_i} \xrightarrow{1} f$. The concatenation of all N automata $\mathcal{A}_{O^{(n)}}$ gives one automaton representing $\prod_{n=1}^{N} p_\lambda(O^{(n)})$. The union of two automata representing functions F_1 and F_2 gives an automaton for the function $F_1 + F_2$. So using unions and concatenations we obtain one AFA (with a single final state) representing the function $F(\lambda, \alpha, \beta, \gamma)$. We can directly construct functional automata for the partial derivatives of F (first order and if needed second order), see Proposition 3.3. Thus we can use a gradient method to find a local extremum of F.

7 Evaluation

In this section we evaluate the quality of a noisy text normalization system that uses the log-linear model presented in Section 2. The system uses a globally convergent variant of the conjugate gradient method, (Gilbert and Nocedal, 1992). The computation of the gradient and the values of the objective function is implemented with functional automata. We test the system on two tasks: (i) OCR-postcorrection of the TREC-5 Confusion Track corpus[1] and (ii) normalization of the 1641 Depositions[2] - a collection of highly non-standard 17th century documents in Early Modern English, (Sweetnam, 2011), digitized at the Trinity College Dublin.

For the task (i) we use a parallel corpus of 30000 training pairs (s, t), where s is a document produced by an OCR system and t is the corrected variant of s. The 30000 pairs were randomly selected from the TREC-5 corpus that has about 5% error on character level. We use 25000 pairs as a training set and the remaining 5000 pairs serve as a test set. With a heruistic dynamic programming algorithm we automatically converted all these 25000 pairs (s, t) into normalization alignments $s \rightarrow t$, see Section 2. We use these alignments to train (a) a candidate generator, (b) smoothed 2-gram language model, to find (c) statistics for the length of the left side of a normalization pair and (d) statistics for normalization pairs with equal left and right sides. Our log-linear model has four feature functions induced by (a), (b), (c) and (d). As a candidate generator we use a variant of the algorithm presented in (Gerdjikov et al., 2013). The word error (WER) rate between s and t in the test set of 5000 pairs is 22.10% and the BLEU (Papineni et al., 2002) is 58.44%. In Table 3 we compare the performace of our log-linear model with four feature functions against a baseline where we use only one feature function, which encodes the candidate generator. Table 3 shows that the combination of the four features reduces more than twice the WER. Precision and recall, obtained on the TREC 5 dataset, for different candidate generators can be found in (Mihov et al., 2007; Schulz et al., 2007; Gerdjikov et al., 2013). To test our system on the task of normalization of the 1641 Depositions, we use a corpus of 500 manually created normalization alignments $s \rightarrow t$, where s is a document in Early Modern English from the 1641 Depositions and t is the normalization of s in contemporary English. We train our system on 450 documents and test it on the other 50. We use five feature functions: (b), (c) and (d) as above and two language models: (a1) one 2-gram language model trained on part of the normalized training documents and (a2) another 2-gram language model trained on large corpus of documents extracted from the entire Gutenberg English language corpus[3]. We obtain WER 5.37% and BLEU 89.34%.

Log-linear model	WER	BLEU
only candidate generator	6.81%	85.24%
candidate generator + language model + other features	3.27%	92.82%

Table 3: Only candidate generator vs. candidate generator + other features. OCR-postcorrection of the TREC-5 corpus.

8 Conclusion

In this paper we considered a general framework for the realization of statistical models. We showed a novel construction proving that the class of functional automata is closed under taking partial derivatives. Thus the functional automata yield efficient training and search procedures using only the usual sum and product operations on real numbers.

We illustrated the power of this mechanism in the cases of CRF's and HMM's, LLM's and n-gram language models. Similar constructions can be applied for the realization of other methods, for example MERT (Och, 2003).

We presented a noisy text normalization system based on functional automata and evaluated its quality.

Acknowledgments

The research work reported in the paper is supported by the CULTURA project, grant 269973, funded by the FP7 Programme (STReP) and Project BG051PO001-3.3.06-0022/19.03.2012.

[1] http://trec.nist.gov/pubs/trec5/t5_proceedings.html
[2] http://1641.tcd.ie
[3] http://www.gutenberg.org

References

Peter F. Brown, Vincent J. Della Pietra, Stephen A. Della Pietra, and Robert L. Mercer, 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2):263–311.

Stanley F. Chen and Joshua Goodman. 1996. An empirical study of smoothing techniques for language modeling. *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, ACL '96, 310–318.

J. N. Darroch and D. Ratcliff. 1972. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480.

Jason Eisner. 2002. Parameter Estimation for Probabilistic Finite-State Transducers. *Proceedings of the 40th annual meeting on Association for Computational Linguistics* ACL '02, 1–8.

Stefan Gerdjikov, Stoyan Mihov, and Vladislav Nenchev. 2013. Extraction of spelling variations from language structure for noisy text correction. *Proceedings of the International Conference on Document Analysis and Recognition*

Jean Charles Gilbert and Jorge Nocedal. 1992. Global Convergence Properties of Conjugate Gradient Methods for Optimization. *SIAM Journal on Optimization*, 2(1):21–42.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company.

B. H. Juang and L. R. Rabiner. 1991. Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3):251–272.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, 48–54

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, 282–289.

Zhifei Li and Jason Eisner 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 40–51.

S. Mihov, P. Mitankin, A. Gotscharek, U. Reffle, C. Schulz, and K. U. Ringlstetter. 2007. Using automated error profiling of texts for improved selection of correction candidates for garbled tokens. *AI 2007: Advances in Artificial Intelligence*, 456-465.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2): 269–311.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, 160–167.

J. Omura. 1967. On the Viterbi decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Translation. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, 311–318.

Lawrence Rabiner. 1989. A tutorial on HMM and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ulrich Reffle. 2011. Efficiently generating correction suggestions for garbled tokens of historical language. *Natural Language Engineering*, 17(02):265–282.

K. U. Schulz, S. Mihov, and P. Mitankin, 2007. Fast selection of small and precise candidate sets from dictionaries for text correction tasks, *Proceedings of the International Conference on Document Analysis and Recognition* 471-475.

Fei Sha and Fernando Pereira, 2003. Shallow parsing with conditional random fields, *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, 134–141.

Mark S. Sweetnam and Barbara A. Fennell. 2011. Natural language processing and early-modern dirty data: applying IBM Languageware to the 1641 depositions. *Literary and Linguistic Computing*, 27(1):39–54