Stochastic Language Generation Using WIDL-expressions and its Application in Machine Translation and Summarization

Radu Soricut
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
radu@isi.edu

Daniel Marcu
Information Sciences Institute
University of Southern California
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
marcu@isi.edu

Abstract

We propose WIDL-expressions as a flexible formalism that facilitates the integration of a generic sentence realization system within end-to-end language processing applications. WIDL-expressions represent compactly probability distributions over finite sets of candidate realizations, and have optimal algorithms for realization via interpolation with language model probability distributions. We show the effectiveness of a WIDL-based NLG system in two sentence realization tasks: automatic translation and headline generation.

1 Introduction

The Natural Language Generation (NLG) community has produced over the years a considerable number of generic sentence realization systems: Penman (Matthiessen and Bateman, 1991), FUF (Elhadad, 1991), Nitrogen (Knight and Hatzivassiloglou, 1995), Fergus (Bangalore and Rambow, 2000), HALogen (Langkilde-Geary, 2002), Amalgam (Corston-Oliver et al., 2002), etc. However, when it comes to end-to-end, text-to-text applications – Machine Translation, Summarization, Question Answering – these generic systems either cannot be employed, or, in instances where they can be, the results are significantly below that of state-of-the-art, application-specific systems (Hajic et al., 2002; Habash, 2003). We believe two reasons explain this state of affairs.

First, these generic NLG systems use input representation languages with complex syntax and semantics. These languages involve deep, semantic-based subject-verb or verb-object relations (such as ACTOR, AGENT, PATIENT, etc., for Penman and FUF), syntactic relations (such as `subject, object, premod`, etc., for HALogen), or lexical dependencies (Fergus, Amalgam). Such inputs cannot be accurately produced by state-of-the-art analysis components from arbitrary textual input in the context of text-to-text applications.

Second, most of the recent systems (starting with Nitrogen) have adopted a hybrid approach to generation, which has increased their robustness. These hybrid systems use, in a first phase, symbolic knowledge to (over)generate a large set of candidate realizations, and, in a second phase, statistical knowledge about the target language (such as stochastic language models) to rank the candidate realizations and find the best scoring one. The disadvantage of the hybrid approach – from the perspective of integrating these systems within end-to-end applications – is that the two generation phases cannot be tightly coupled. More precisely, input-driven preferences and target language–driven preferences cannot be integrated in a true probabilistic model that can be trained and tuned for maximum performance.

In this paper, we propose WIDL-expressions (WIDL stands for Weighted Interleave, Disjunction, and Lock, after the names of the main operators) as a representation formalism that facilitates the integration of a generic sentence realization system within end-to-end language applications. The WIDL formalism, an extension of the IDL-expressions formalism of Nederhof and Satta (2004), has several crucial properties that differentiate it from previously-proposed NLG representation formalisms. First, it has a simple syntax (expressions are built using four operators) and a simple, formal semantics (probability distributions over finite sets of strings). Second, it is a compact representation that grows linearly

Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pages 1105–1112,
Sydney, July 2006.

in the number of words available for generation (see Section 2). (In contrast, representations such as word lattices (Knight and Hatzivassiloglou, 1995) or non-recursive CFGs (Langkilde-Geary, 2002) require exponential space in the number of words available for generation (Nederhof and Satta, 2004).) Third, it has good computational properties, such as optimal algorithms for intersection with n-gram language models (Section 3). Fourth, it is flexible with respect to the amount of linguistic processing required to produce WIDL-expressions directly from text (Sections 4 and 5). Fifth, it allows for a tight integration of input-specific preferences and target-language preferences via interpolation of probability distributions using log-linear models. We show the effectiveness of our proposal by directly employing a generic WIDL-based generation system in two end-to-end tasks: machine translation and automatic headline generation.

2 The WIDL Representation Language

2.1 WIDL-expressions

In this section, we introduce WIDL-expressions, a formal language used to *compactly* represent probability distributions over finite sets of strings.

Given a finite alphabet of symbols Σ, atomic WIDL-expressions are of the form a, with $a \in \Sigma$. For a WIDL-expression $\omega = a$, its semantics is a probability distribution $\sigma_{\text{widl}}(\omega) : \mathsf{Dom}_\omega \rightarrow [0,1]$, where $\mathsf{Dom}_\omega = \{a\}$ and $\sigma_{\text{widl}}(\omega)(a) = 1$. Complex WIDL-expressions are created from other WIDL-expressions, by employing the following four operators, as well as operator distribution functions δ_i from an alphabet Δ.

Weighted Disjunction. If $\omega_1, \ldots, \omega_n$ are WIDL-expressions, then $\omega = \vee_{\delta_0}(\omega_1, \ldots, \omega_n)$, with $\delta_0 : \{1, \ldots, n\} \rightarrow [0,1]$, specified such that $\Sigma_{k \in \mathsf{dom}(\delta_0)} \delta_0(k) = 1$, is a WIDL-expression. Its semantics is a probability distribution $\sigma_{\text{widl}}(\omega) : \mathsf{Dom}_\omega \rightarrow [0,1]$, where $\mathsf{Dom}_\omega = \bigcup_{i=1}^{n} \mathsf{Dom}_{\omega_i}$, and the probability values are induced by δ_0 and $\sigma_{\text{widl}}(\omega_i)$, $1 \leq i \leq n$. For example, if $\omega = \vee_{\delta_0}(\mathrm{a}, \mathrm{b})$, $\delta_0 = \{1 \rightarrow 0.8,\ 2 \rightarrow 0.2\}$, its semantics is a probability distribution $\sigma_{\text{widl}}(\omega)$ over $\mathsf{Dom}_\omega = \{\mathrm{a}, \mathrm{b}\}$, defined by $\sigma_{\text{widl}}(\omega)(\mathrm{a}) = \delta_0(1) = 0.8$ and $\sigma_{\text{widl}}(\omega)(\mathrm{b}) = \delta_0(2) = 0.2$.

Precedence. If ω_1, ω_2 are WIDL-expressions, then $\omega = \omega_1 \cdot \omega_2$ is a WIDL-expression. Its semantics is a probability distribution $\sigma_{\text{widl}}(\omega) : \mathsf{Dom}_\omega \rightarrow [0,1]$, where Dom_ω is the set of all strings that obey the precedence imposed over the arguments, and the probability values are induced by $\sigma_{\text{widl}}(\omega_1)$ and $\sigma_{\text{widl}}(\omega_2)$. For example, if $\omega_1 = \vee_{\delta_1}(\mathrm{a}, \mathrm{b})$, $\delta_1 = \{1 \rightarrow 0.8,\ 2 \rightarrow 0.2\}$, and $\omega_2 = \vee_{\delta_2}(\mathrm{c}, \mathrm{d})$, $\delta_2 = \{1 \rightarrow 0.6,\ 2 \rightarrow 0.4\}$, then $\omega = \omega_1 \cdot \omega_2$ represents a probability distribution $\sigma_{\text{widl}}(\omega)$ over the set $\mathsf{Dom}_\omega = \{\mathrm{ac}, \mathrm{ad}, \mathrm{bc}, \mathrm{bd}\}$, defined by $\sigma_{\text{widl}}(\omega)(\mathrm{ac}) = \delta_1(1)\delta_2(1) = 0.48$, $\sigma_{\text{widl}}(\omega)(\mathrm{ad}) = \delta_1(1)\delta_2(2) = 0.32$, etc.

Weighted Interleave. If $\omega_1, \ldots, \omega_n$ are WIDL-expressions, then $\omega = \|_{\delta_0}(\omega_1, \omega_2, \ldots, \omega_n)$, with $\delta_0 : S \cup \{\mathsf{other\ perms}\} \cup \{\mathsf{shuffles}\} \rightarrow [0,1]$, $S \subseteq \mathsf{Perm}_n$, specified such that $\Sigma_{a \in \mathsf{dom}(\delta_0)} \delta_0(a) = 1$, is a WIDL-expression. Its semantics is a probability distribution $\sigma_{\text{widl}}(\omega) : \mathsf{Dom}_\omega \rightarrow [0,1]$, where Dom_ω consists of all the possible interleavings of strings from Dom_{ω_i}, $1 \leq i \leq n$, and the probability values are induced by δ_0 and $\sigma_{\text{widl}}(\omega_i)$. The distribution function δ_0 is defined either explicitly, over $S \subseteq \mathsf{Perm}_n$ (the set of all permutations of n elements), or implicitly, as $\delta_0(\mathsf{other\ perms})$. Because the set of argument permutations is a subset of all possible interleavings, δ_0 also needs to specify the probability mass for the strings that are not argument permutations, $\delta_0(\mathsf{shuffles})$. For example, if $\omega = \|_{\delta_0}(\mathrm{a} \cdot \mathrm{b}, \mathrm{c})$, $\delta_0 = \{1\ 2 \rightarrow 0.80,\ \mathsf{other\ perms} \overset{\mathsf{uniform}}{\rightarrow} 0.15, \mathsf{shuffles} \overset{\mathsf{uniform}}{\rightarrow} 0.05\}$, its semantics is a probability distribution $\sigma_{\text{widl}}(\omega)$, with domain $\mathsf{Dom}_\omega = \{\mathrm{abc}, \mathrm{cab}, \mathrm{acb}\}$, defined by $\sigma_{\text{widl}}(\omega)(\mathrm{abc}) = \delta_0(1\ 2) = 0.80$, $\sigma_{\text{widl}}(\omega)(\mathrm{cab}) = \frac{\delta_0(\mathsf{other\ perms})}{1} = 0.15$, $\sigma_{\text{widl}}(\omega)(\mathrm{acb}) = \frac{\delta_0(\mathsf{shuffles})}{1} = 0.05$.

Lock. If ω' is a WIDL-expression, then $\omega = \times(\omega')$ is a WIDL-expression. The semantic mapping $\sigma_{\text{widl}}(\omega)$ is the same as $\sigma_{\text{widl}}(\omega')$, except that Dom_ω contains strings in which no additional symbol can be interleaved. For example, if $\omega = \|_{\delta_0}(\times(\mathrm{a} \cdot \mathrm{b}), \mathrm{c})$, $\delta_0 = \{1\ 2 \rightarrow 0.80,\ \mathsf{other\ perms} \rightarrow 0.20\}$, its semantics is a probability distribution $\sigma_{\text{widl}}(\omega)$, with domain $\mathsf{Dom}_\omega = \{\mathrm{cab}, \mathrm{abc}\}$, defined by $\sigma_{\text{widl}}(\omega)(\mathrm{abc}) = \delta_0(1\ 2) = 0.80$, $\sigma_{\text{widl}}(\omega)(\mathrm{cab}) = \frac{\delta_0(\mathsf{other\ perms})}{1} = 0.20$.

In Figure 1, we show a more complex WIDL-expression. The probability distribution δ_1 associated with the operator $\|_{\delta_1}$ assigns probability 0.2 to the argument order 2 1 3; from a probability mass of 0.7, it assigns uniformly, for each of the remaining $3! - 1 = 5$ argument permutations, a permutation probability value of $\frac{0.7}{5} = 0.14$. The

$$\|_{\delta_1}(\times(\mathsf{turkish}\cdot\mathsf{government}), \vee_{\delta_2}(\times(\mathsf{rebels}\cdot\mathsf{fighting}), \times(\mathsf{attacked}\cdot\mathsf{rebels})), \mathsf{in}\cdot\mathsf{iraq}),$$
$$\delta_1 = \{2\,1\,3 \rightarrow 0.2,\ \mathsf{other\ perms} \overset{\mathsf{uniform}}{\rightarrow} 0.7, \mathsf{shuffles} \overset{\mathsf{uniform}}{\rightarrow} 0.1\},\ \delta_2 = \{1 \rightarrow 0.65,\ 2 \rightarrow 0.35\}$$

Figure 1: An example of a WIDL-expression.

remaining probability mass of 0.1 is left for the 12 shuffles associated with the unlocked expression $\mathsf{in}\cdot\mathsf{iraq}$, for a shuffle probability of $\frac{0.1}{12} = 0.008$. The list below enumerates some of the $\langle \mathsf{string}, p(\mathsf{string})\rangle$ pairs that belong to the probability distribution defined by our example:

rebels fighting turkish government in iraq	0.130
in iraq attacked rebels turkish goverment	0.049
in turkish goverment iraq rebels fighting	0.005

The following result characterizes an important representation property for WIDL-expressions.

Theorem 1 *A WIDL-expression ω over Σ and Δ using n atomic expressions has space complexity O(n), if the operator distribution functions of ω have space complexity at most O(n).*

For proofs and more details regarding WIDL-expressions, we refer the interested reader to (Soricut, 2006). Theorem 1 ensures that high-complexity hypothesis spaces can be represented efficiently by WIDL-expressions (Section 5).

2.2 WIDL-graphs and Probabilistic Finite-State Acceptors

WIDL-graphs. Equivalent at the representation level with WIDL-expressions, WIDL-graphs allow for formulations of algorithms that process them. For each WIDL-expression ω, there exists an equivalent WIDL-graph G_ω. As an example, we illustrate in Figure 2(a) the WIDL-graph corresponding to the WIDL-expression in Figure 1. WIDL-graphs have an initial vertex v_s and a final vertex v_e. Vertices v_0, v_6, and v_{20} with in-going edges labeled $\vdash^1_{\delta_1}$, $\vdash^2_{\delta_1}$, and $\vdash^3_{\delta_1}$, respectively, and vertices v_5, v_{19}, and v_{23} with out-going edges labeled $\dashv^1_{\delta_1}$, $\dashv^2_{\delta_1}$, and $\dashv^3_{\delta_1}$, respectively, result from the expansion of the $\|_{\delta_1}$ operator. Vertices v_7 and v_{13} with in-going edges labeled $(^1_{\delta_2}$, $(^2_{\delta_2}$, respectively, and vertices v_{12} and v_{18} with out-going edges labeled $)^1_{\delta_2}$, $)^2_{\delta_2}$, respectively, result from the expansion of the $\vee_{\delta_2}$ operator.

With each WIDL-graph G_ω, we associate a probability distribution. The domain of this distribution is the finite collection of strings that can be generated from the paths of a WIDL-specific traversal of G_ω, starting from v_s and ending in v_e. Each path (and its associated string) has a probability value induced by the probability distribution functions associated with the edge labels of G_ω. A WIDL-expression ω and its corresponding WIDL-graph G_ω are said to be equivalent because they represent the same distribution $\sigma_{\mathrm{widl}}(\omega)$.

WIDL-graphs and Probabilistic FSA. Probabilistic finite-state acceptors (pFSA) are a well-known formalism for representing probability distributions (Mohri et al., 2002). For a WIDL-expression ω, we define a mapping, called UNFOLD, between the WIDL-graph G_ω and a pFSA A_ω. A state s in A_ω is created for each set of WIDL-graph vertices that can be reached simultaneously when traversing the graph. State s records, in what we call a $\|$-stack (interleave stack), the order in which $\vdash^i_\delta$,$\dashv^i_\delta$–bordered sub-graphs are traversed. Consider Figure 2(b), in which state $[v_0 v_9 v_{23}, \{\langle_{\delta_1} 3\,2\}]$ (at the bottom) corresponds to reaching vertices v_0, v_9, and v_{23} (see the WIDL-graph in Figure 2(a)), by first reaching vertex v_{23} (inside the $\vdash^3_{\delta_1}$, $\dashv^3_{\delta_1}$–bordered sub-graph), and then reaching vertex v_9 (inside the $\vdash^2_{\delta_1}$, $\dashv^2_{\delta_1}$–bordered sub-graph).

A transition labeled a between two A_ω states s_1 and s_2 in A_ω exists if there exists a vertex v_j in the description of s_1 and a vertex v_k in the description of s_2 such that there exists a path in G_ω between v_j and v_k, and a is the only Σ-labeled transitions in this path. For example, transition $[v_0 v_9 v_{23}, \{\langle_{\delta_1} 3\,2\}] \overset{\mathsf{rebels}}{\rightarrow} [v_0 v_{19} v_{23}, \{\langle_{\delta_1} 3\,2\}]$ (Figure 2(b)) results from unfolding the path $v_9 \overset{\varepsilon}{\rightarrow} v_{10} \overset{\mathsf{rebels}}{\rightarrow} v_{11} \overset{\varepsilon}{\rightarrow} v_{12} \overset{)^1_{\delta_2}}{\rightarrow} v_{19}$ (Figure 2(a)). A transition labeled ε between two A_ω states s_1 and s_2 in A_ω exists if there exists a vertex v_j in the description of s_1 and vertices $v^1_k, \ldots, v^n_k$ in the description of s_2, such that $v_j \overset{\vdash^i_\delta}{\rightarrow} v^i_k \in G_\omega$, $1 \leq i \leq n$ (see transition $[v_s,] \overset{\varepsilon}{\rightarrow} [v_0 v_6 v_{20}, \{\langle_{\delta_1}\rangle_{\delta_1}\}]$), or if there exists vertices $v^1_j, \ldots, v^n_j$ in the description of s_1 and vertex v_k in the description of s_2, such that $v^i_j \overset{\dashv^i_\delta}{\rightarrow} v_k \in G_\omega$, $1 \leq i \leq n$. The ε-transitions

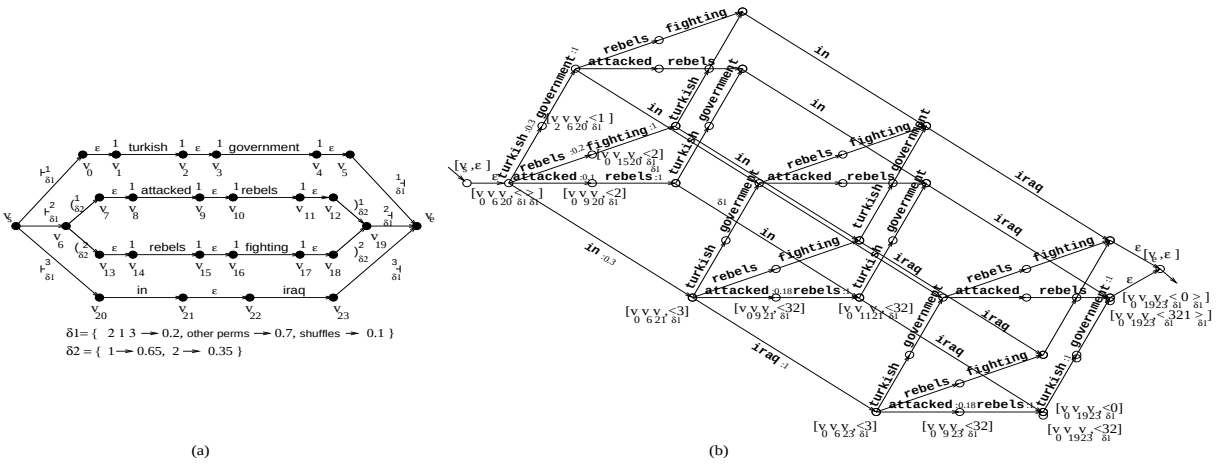

Figure 2: The WIDL-graph corresponding to the WIDL-expression in Figure 1 is shown in (a). The probabilistic finite-state acceptor (pFSA) that corresponds to the WIDL-graph is shown in (b).

are responsible for adding and removing, respectively, the $\langle_\delta$,$\rangle_\delta$ symbols in the $\|$-stack. The probabilities associated with A_ω transitions are computed using the vertex set and the $\|$-stack of each A_ω state, together with the distribution functions of the $\vee$ and $\|$ operators. For a detailed presentation of the UNFOLD relation we refer the reader to (Soricut, 2006).

3 Stochastic Language Generation from WIDL-expressions

3.1 Interpolating Probability Distributions in a Log-linear Framework

Let us assume a finite set E of strings over a finite alphabet Σ, representing the set of possible sentence realizations. In a log-linear framework, we have a vector of feature functions $\overline{h} = \langle h_0 h_1 \ldots h_M \rangle$, and a vector of parameters $\overline{\lambda} = \langle \lambda_0 \lambda_1 \ldots \lambda_M \rangle$. For any $e \in E$, the interpolated probability $P(e)$ can be written under a log-linear model as in Equation 1:

$$P(e) = \frac{\exp[\Sigma_{m=0}^{M} \lambda_m h_m(e)]}{\Sigma_{e'} \exp[\Sigma_{m=0}^{M} \lambda_m h_m(e')]} \quad (1)$$

We can formulate the search problem of finding the most probable realization e under this model as shown in Equation 2, and therefore we do not need to be concerned about computing expensive normalization factors.

$$\arg\max_e P(e) = \arg\max_e \exp[\Sigma_{m=0}^{M} \lambda_m h_m(e)] \quad (2)$$

For a given WIDL-expression ω over Σ, the set E is defined by $\mathsf{dom}(\sigma_{\text{widl}}(\omega))$, and feature function h_0 is taken to be $\sigma_{\text{widl}}(\omega)$. Any language model we want to employ may be added in Equation 2 as a feature function h_i, $i \geq 1$.

3.2 Algorithms for Intersecting WIDL-expressions with Language Models

Algorithm WIDL-NGLM-A* (Figure 3) solves the search problem defined by Equation 2 for a WIDL-expression ω (which provides feature function h_0) and M n-gram language models (which provide feature functions $h_1, \ldots, h_M$). It does so by incrementally computing UNFOLD for G_ω (i.e., on-demand computation of the corresponding pFSA A_ω), by keeping track of a set of active states, called $active$. The set of newly UNFOLDed states is called $unfold$. Using Equation 1 (unnormalized), we EVALUATE the current $P(e)$ scores for the $unfold$ states. Additionally, EVALUATE uses an admissible heuristic function to compute future (admissible) scores for the $unfold$ states.

The algorithm PUSHes each state from the current $unfold$ into a priority queue Q, which sorts the states according to their total score (current + admissible). In the next iteration, $active$ is a singleton set containing the state POPed out from the top of Q. The admissible heuristic function we use is the one defined in (Soricut and Marcu, 2005), using Equation 1 (unnormalized) for computing the event costs. Given the existence of the admissible heuristic and the monotonicity property of the unfolding provided by the priority queue Q, the proof for A* optimality (Russell and Norvig, 1995) guarantees that WIDL-NGLM-A* finds a path in A_w that provides an optimal solution.

WIDL-NGLM-A$^*(G_\omega, \overline{h}, \overline{\lambda})$

1 $active \leftarrow \{[v_s, \{\}]\}$
2 $flag \leftarrow 1$
3 **while** $flag$
4 **do** $unfold \leftarrow$ UNFOLD$(G_\omega, active)$
5 EVALUATE$(unfold, \overline{h}, \overline{\lambda})$
6 **if** $active = \{[v_e, \{\}]\}$
7 **then** $flag \leftarrow 0$
8 **for** each $state$ in $unfold$
do PUSH$(Q, state)$
$active \leftarrow$ POP(Q)
9 **return** $active$

Figure 3: A* algorithm for interpolating WIDL-expressions with n-gram language models.

An important property of the WIDL-NGLM-A* algorithm is that the UNFOLD relation (and, implicitly, the A_ω acceptor) is computed only partially, for those states for which the total cost is less than the cost of the optimal path. This results in important savings, both in space and time, over simply running a single-source shortest-path algorithm for directed acyclic graphs (Cormen et al., 2001) over the full acceptor A_ω (Soricut and Marcu, 2005).

4 Headline Generation using WIDL-expressions

We employ the WIDL formalism (Section 2) and the WIDL-NGLM-A* algorithm (Section 3) in a summarization application that aims at producing both informative and fluent headlines. Our headlines are generated in an abstractive, bottom-up manner, starting from words and phrases. A more common, extractive approach operates top-down, by starting from an extracted sentence that is compressed (Dorr et al., 2003) and annotated with additional information (Zajic et al., 2004).

Automatic Creation of WIDL-expressions for Headline Generation. We generate WIDL-expressions starting from an input document. First, we extract a weighted list of topic keywords from the input document using the algorithm of Zhou and Hovy (2003). This list is enriched with phrases created from the lexical dependencies the topic keywords have in the input document. We associate probability distributions with these phrases using their frequency (we assume that higher frequency is indicative of increased importance) and their position in the document (we assume that proximity to the beginning of the document is also indicative of importance). In Figure 4, we present an example of input keywords and lexical-dependency phrases automatically extracted from a document describing incidents at the Turkey-Iraq border.

Keywords {iraq 0.32, syria 0.25, rebels 0.22,
kurdish 0.17, turkish 0.14, attack 0.10}
Phrases
iraq {in iraq 0.4, northern iraq 0.5,iraq and iran 0.1},
syria { into syria 0.6, and syria 0.4 }
rebels { attacked rebels 0.7,rebels fighting 0.3}
...
↓ WIDL-expression & trigram interpolation
TURKISH GOVERNMENT ATTACKED REBELS IN IRAQ AND SYRIA

Figure 4: Input and output for our automatic headline generation system.

The algorithm for producing WIDL-expressions combines the lexical-dependency phrases for each keyword using a $\vee$ operator with the associated probability values for each phrase multiplied with the probability value of each topic keyword. It then combines all the $\vee$-headed expressions into a single WIDL-expression using a $\|$ operator with uniform probability. The WIDL-expression in Figure 1 is a (scaled-down) example of the expressions created by this algorithm. On average, a WIDL-expression created by this algorithm, using $m = 6$ keywords and an average of $k = 4$ lexical-dependency phrases per keyword, compactly encodes a candidate set of about 3 million possible realizations. As the specification of the $\|_\delta$ operator takes space $O(1)$ for uniform δ, Theorem 1 guarantees that the space complexity of these expressions is $O(mk)$.

Finally, we generate headlines from WIDL-expressions using the WIDL-NGLM-A* algorithm, which interpolates the probability distributions represented by the WIDL-expressions with n-gram language model distributions. The output presented in Figure 4 is the most likely headline realization produced by our system.

Headline Generation Evaluation. To evaluate the accuracy of our headline generation system, we use the documents from the DUC 2003 evaluation competition. Half of these documents are used as development set (283 documents),

ALG	♯(uni)	♯(bi)	Len.	Rouge_1	$\textbf{Rouge}_2$
Extractive					
Lead10	458	114	9.9	20.8	11.1
HedgeTrimmer†	399	104	7.4	18.1	9.9
Topiary‡	576	115	9.9	26.2	12.5
Abstractive					
Keywords	585	22	9.9	26.6	5.5
Webcl	311	76	7.3	14.1	7.5
WIDL-A*	562	126	10.0	25.5	**12.9**

Table 1: Headline generation evaluation. We compare extractive algorithms against abstractive algorithms, including our WIDL-based algorithm.

and the other half is used as test set (273 documents). We automatically measure performance by comparing the produced headlines against one reference headline produced by a human using ROUGE_2 (Lin, 2004).

For each input document, we train two language models, using the SRI Language Model Toolkit (with modified Kneser-Ney smoothing). A general trigram language model, trained on 170M English words from the Wall Street Journal, is used to model fluency. A document-specific trigram language model, trained on-the-fly for each input document, accounts for both fluency and content validity. We also employ a word-count model (which counts the number of words in a proposed realization) and a phrase-count model (which counts the number of phrases in a proposed realization), which allow us to learn to produce headlines that have restrictions in the number of words allowed (10, in our case). The interpolation weights $\overline{\lambda}$ (Equation 2) are trained using discriminative training (Och, 2003) using ROUGE_2 as the objective function, on the development set.

The results are presented in Table 1. We compare the performance of several extractive algorithms (which operate on an extracted sentence to arrive at a headline) against several abstractive algorithms (which create headlines starting from scratch). For the extractive algorithms, **Lead10** is a baseline which simply proposes as headline the lead sentence, cut after the first 10 words. **HedgeTrimmer**† is our implementation of the Hedge Trimer system (Dorr et al., 2003), and **Topiary**‡ is our implementation of the Topiary system (Zajic et al., 2004). For the abstractive algorithms, **Keywords** is a baseline that proposes as headline the sequence of topic keywords, **Webcl** is the system

THREE GORGES PROJECT IN CHINA HAS WON APPROVAL
WATER IS LINK BETWEEN CLUSTER OF E. COLI CASES
SRI LANKA 'S JOINT VENTURE TO EXPAND EXPORTS
OPPOSITION TO EUROPEAN UNION SINGLE CURRENCY EURO
OF INDIA AND BANGLADESH WATER BARRAGE

Figure 5: Headlines generated automatically using a WIDL-based sentence realization system.

described in (Zhou and Hovy, 2003), and **WIDL-A*** is the algorithm described in this paper.

This evaluation shows that our WIDL-based approach to generation is capable of obtaining headlines that compare favorably, in both content and fluency, with extractive, state-of-the-art results (Zajic et al., 2004), while it outperforms a previously-proposed abstractive system by a wide margin (Zhou and Hovy, 2003). Also note that our evaluation makes these results directly comparable, as they use the same parsing and topic identification algorithms. In Figure 5, we present a sample of headlines produced by our system, which includes both good and not-so-good outputs.

5 Machine Translation using WIDL-expressions

We also employ our WIDL-based realization engine in a machine translation application that uses a two-phase generation approach: in a first phase, WIDL-expressions representing large sets of possible translations are created from input foreign-language sentences. In a second phase, we use our generic, WIDL-based sentence realization engine to intersect WIDL-expressions with an n-gram language model. In the experiments reported here, we translate between Chinese (source language) and English (target language).

Automatic Creation of WIDL-expressions for MT. We generate WIDL-expressions from Chinese strings by exploiting a phrase-based translation table (Koehn et al., 2003). We use an algorithm resembling probabilistic bottom-up parsing to build a WIDL-expression for an input Chinese string: each contiguous span (i, j) over a Chinese string $C_{i,j}$ is considered a possible "constituent", and the "non-terminals" associated with each constituent are the English phrase translations $E^k_{i,j}$ that correspond in the translation table to the Chinese string $C_{i,j}$. Multiple-word English phrases, such as $\mathsf{w}_1\,\mathsf{w}_2\,\mathsf{w}_3$, are represented as WIDL-expressions using the precedence $(\cdot)$ and

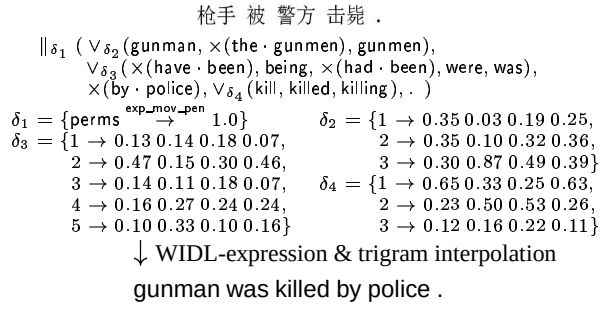

Figure 6: A Chinese string is converted into a WIDL-expression, which provides a translation as the best scoring hypothesis under the interpolation with a trigram language model.

lock (×) operators, as $\times(\mathsf{w}_1 \cdot \mathsf{w}_2 \cdot \mathsf{w}_3)$. To limit the number of possible translations $E^k_{i,j}$ corresponding to a Chinese span $C_{i,j}$, we use a probabilistic beam b and a histogram beam s to beam out low probability translation alternatives. At this point, each $C_{i,j}$ span is "tiled" with likely translations $E^k_{i,j}$ taken from the translation table.

Tiles that are adjacent are joined together in a larger tile by a $\|_\delta$ operator, where $\delta = \{\text{perms} \overset{\text{exp\_mov\_pen}}{\rightarrow} 1\}$. That is, reordering of the component tiles are permitted by the $\|_\delta$ operators (assigned non-zero probability), but the longer the movement from the original order of the tiles, the lower the probability. (This distortion model is similar with the one used in (Koehn, 2004).) When multiple tiles are available for the same span (i, j), they are joined by a $\vee_\delta$ operator, where δ is specified by the probability distributions specified in the translation table. Usually, statistical phrase-based translation tables specify not only one, but multiple distributions that account for context preferences. In our experiments, we consider four probability distributions: $p(\overline{f}|\overline{e}), p(\overline{e}|\overline{f}), p_{lex}(\overline{f}|\overline{e})$, and $p_{lex}(\overline{e}|\overline{f})$, where $\overline{f}$ and $\overline{e}$ are Chinese-English phrase translations as they appear in the translation table. In Figure 6, we show an example of WIDL-expression created by this algorithm[1].

On average, a WIDL-expression created by this algorithm, using an average of $m = 38$ tiles per sentence (for an average input sentence length of 30 words) and an average of $k = 9$ possible translations per tile, encodes a candidate set of about 10^{50} possible translations. As the specification of the $\|_\delta$ operators takes space $O(1)$, Theorem 1 guarantees that these WIDL-expressions encode compactly these huge spaces in $O(mk)$.

In the second phase, we employ our WIDL-based realization engine to interpolate the distribution probabilities of WIDL-expressions with a trigram language model. In the notation of Equation 2, we use four feature functions $h_0, \ldots, h_3$ for the WIDL-expression distributions (one for each probability distribution encoded); a feature function h_4 for a trigram language model; a feature function h_5 for a word-count model, and a feature function h_6 for a phrase-count model.

As acknowledged in the Machine Translation literature (Germann et al., 2003), full A* search is not usually possible, due to the large size of the search spaces. We therefore use an approximation algorithm, called WIDL-NGLM-A$^*_k$, which considers for unfolding only the nodes extracted from the priority queue Q which already unfolded a path of length greater than or equal to the maximum length already unfolded minus k (we used $k = 2$ in the experiments reported here).

MT Performance Evaluation. When evaluated against the state-of-the-art, phrase-based decoder Pharaoh (Koehn, 2004), using the same experimental conditions – translation table trained on the FBIS corpus (7.2M Chinese words and 9.2M English words of parallel text), trigram language model trained on 155M words of English newswire, interpolation weights $\overline{\lambda}$ (Equation 2) trained using discriminative training (Och, 2003) (on the 2002 NIST MT evaluation set), probabilistic beam b set to 0.01, histogram beam s set to 10 – and BLEU (Papineni et al., 2002) as our metric, the WIDL-NGLM-A$^*_2$ algorithm produces translations that have a BLEU score of 0.2570, while Pharaoh translations have a BLEU score of 0.2635. The difference is not statistically significant at 95% confidence level.

These results show that the WIDL-based approach to machine translation is powerful enough to achieve translation accuracy comparable with state-of-the-art systems in machine translation.

6 Conclusions

The approach to sentence realization we advocate in this paper relies on WIDL-expressions, a formal language with convenient theoretical properties that can accommodate a wide range of generation scenarios. In the worst case, one can work with simple bags of words that encode no context

[1] English reference: the gunman was shot dead by the police.

preferences (Soricut and Marcu, 2005). One can also work with bags of words and phrases that encode context preferences, a scenario that applies to current approaches in statistical machine translation (Section 5). And one can also encode context and ordering preferences typically used in summarization (Section 4).

The generation engine we describe enables a tight coupling of content selection with sentence realization preferences. Its algorithm comes with theoretical guarantees about its optimality. Because the requirements for producing WIDL-expressions are minimal, our WIDL-based generation engine can be employed, with state-of-the-art results, in a variety of text-to-text applications.

Acknowledgments This work was partially supported under the GALE program of the Defense Advanced Research Projects Agency, Contract No. HR0011-06-C-0022.

References

Srinivas Bangalore and Owen Rambow. 2000. Using TAG, a tree model, and a language model for generation. In *Proceedings of the Fifth International Workshop on Tree-Adjoining Grammars (TAG+)*.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*. The MIT Press and McGraw-Hill.

Simon Corston-Oliver, Michael Gamon, Eric K. Ringger, and Robert Moore. 2002. An overview of Amalgam: A machine-learned generation module. In *Proceedings of the INLG*.

Bonnie Dorr, David Zajic, and Richard Schwartz. 2003. Hedge trimmer: a parse-and-trim approach to headline generation. In *Proceedings of the HLT-NAACL Text Summarization Workshop*, pages 1–8.

Michael Elhadad. 1991. FUF User manual — version 5.0. Technical Report CUCS-038-91, Department of Computer Science, Columbia University.

Ulrich Germann, Mike Jahr, Kevin Knight, Daniel Marcu, and Kenji Yamada. 2003. Fast decoding and optimal decoding for machine translation. *Artificial Intelligence*, 154(1–2):127-143.

Nizar Habash. 2003. Matador: A large-scale Spanish-English GHMT system. In *Proceedings of AMTA*.

J. Hajic, M. Cmejrek, B. Dorr, Y. Ding, J. Eisner, D. Gildea, T. Koo, K. Parton, G. Penn, D. Radev, and O. Rambow. 2002. Natural language generation in the context of machine translation. Summer workshop final report, Johns Hopkins University.

K. Knight and V. Hatzivassiloglou. 1995. Two level, many-path generation. In *Proceedings of the ACL*.

Philipp Koehn, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proceedings of the HLT-NAACL*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine transltion models. In *Proceedings of the AMTA*, pages 115–124.

I. Langkilde-Geary. 2002. *A foundation for general-purpose natural language generation: sentence realization using probabilistic models of language*. Ph.D. thesis, University of Southern California.

Chin-Yew Lin. 2004. ROUGE: a package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*.

Christian Matthiessen and John Bateman. 1991. *Text Generation and Systemic-Functional Linguistic*. Pinter Publishers, London.

Mehryar Mohri, Fernando Pereira, and Michael Riley. 2002. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88.

Mark-Jan Nederhof and Giorgio Satta. 2004. IDL-expressions: a formalism for representing and parsing finite languages in natural language processing. *Journal of Artificial Intelligence Research*, pages 287–317.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the ACL*, pages 160–167.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *In Proceedings of the ACL*, pages 311–318.

Stuart Russell and Peter Norvig. 1995. *Artificial Intelligence. A Modern Approach*. Prentice Hall.

Radu Soricut and Daniel Marcu. 2005. Towards developing generation algorithms for text-to-text applications. In *Proceedings of the ACL*, pages 66–74.

Radu Soricut. 2006. *Natural Language Generation for Text-to-Text Applications Using an Information-Slim Representation*. Ph.D. thesis, University of Southern California.

David Zajic, Bonnie J. Dorr, and Richard Schwartz. 2004. BBN/UMD at DUC-2004: Topiary. In *Proceedings of the NAACL Workshop on Document Understanding*, pages 112–119.

Liang Zhou and Eduard Hovy. 2003. Headline summarization at ISI. In *Proceedings of the NAACL Workshop on Document Understanding*.