Hierarchical Statistical Semantic Realization for Minimal Recursion Semantics

Matic Horvat
Computer Laboratory
University of Cambridge
mh693@cam.ac.uk

Ann Copestake
Computer Laboratory
University of Cambridge
aac10@cam.ac.uk

William Byrne
Department of Engineering
University of Cambridge
wjb31@cam.ac.uk

Abstract

We introduce a robust statistical approach to realization from Minimal Recursion Semantics representations. The approach treats realization as a translation problem, transforming the Dependency MRS graph representation to a surface string. Translation is based on a Synchronous Context-Free Grammar that is automatically extracted from a large corpus of parsed sentences. We have evaluated the new approach on the Wikiwoods corpus, where it shows promising results.[1]

1 Introduction

Realization from Minimal Recursion Semantics (MRS) representations has traditionally used a chart-based approach governed by a resource grammar. Introduced by Carroll et al. (1999) and Carroll and Oepen (2005), the chart-based approach is lexically-driven and is able to produce a large number of candidate surface strings which may be ranked using an N-gram language model or using discriminative machine learning (Velldal and Oepen, 2005; Velldal, 2009).

As the chart-based realization relies on a resource grammar, it tends to perform well when realizing from MRS representations that were created by a parser using the same resource grammar. However, the chart-based approach may fail to produce any output when the MRS representation has missing or incorrect parts. This is a significant issue for the MRS representations produced as a result of semantic transfer in semantic transfer translation systems such as LOGON (Lø nning et al., 2004) due to the difficulty of the translation problem. Consequently, the realization component is unable to produce any output and, in turn, translation fails.

In this paper we describe a first attempt at statistical realization from MRS representations. The approach treats realization as a translation problem, transforming the Dependency MRS graph representation to a surface string. The approach draws inspiration from Statistical Machine Translation, namely the hierarchical phrase-based approach to translation introduced by Chiang (2005, 2007). We will refer to the new approach as Hierarchical Statistical Semantic Realization or HSSR.

As part of the HSSR system, we present an approach for the automatic extraction of salient hierarchical rules for realization. The approach creates rules by considering DMRS subgraphs and corresponding surface substrings. The rules are created in two stages, first creating terminal rules, followed by nonterminal rules. The latter are created by 'subtracting' terminal rules from each other. The realization rules are extracted from a large parsed corpus to form a Synchronous Context-Free Grammar (SCFG).

We build on the ideas behind HiFST, a hierarchical phrase-based decoder introduced by Iglesias et al. (2009), to create a realization decoder. The decoder represents realization rules as Weighted Finite State Acceptors (WFSA). It uses WFSA operations to create a lattice encoding all possible realizations under a given SCFG. An N-gram language model is applied to the lattice to encourage fluency in surface realizations. The best realization is selected by finding the shortest path through the lattice.

[1]This research was partially supported by Qualcomm Research Scholarship and Churchill College Scholarship. The authors would also like to thank Juan Pino and Aurelien Waite for their help with software and experiments.

Proceedings of the 11th International Conference on Computational Semantics, pages 107–117,
London, UK, April 15-17 2015. ©2015 Association for Computational Linguistics

The long term goal of the HSSR system is to provide a robust alternative to chart-based realization that would be especially useful for realization in semantic transfer-based translation systems. However, in this paper we focus on presenting the main ideas behind HSSR and not on providing a direct alternative to the traditional approach. The system in its current stage of development lacks maturity and efficiency required by its potential applications. Consequently, we make some simplifying assumptions during evaluation, which we discuss in the relevant parts of the paper.

The HSSR approach is suitable for realization in any language, provided that there is a resource grammar of the language available. We evaluated its performance on the Wikiwoods corpus (Flickinger et al., 2010), a large deep parsed corpus of English Wikipedia which provides a large collection of MRS representations aligned with surface realizations that are suitable for learning the realization grammar.

We measure the performance of the HSSR system using BLEU and discuss its strengths and weakness using example output. The system shows promising results, with the main issues stemming from the lack of efficiency.

2 Minimal Recursion Semantics

Minimal Recursion Semantics (MRS) is a framework for computational semantics introduced by Copestake et al. (1995) and formally described in Copestake et al. (2005). As discussed there, MRS is a metalanguage for describing semantic structures in some underlying object language: the object language usually discussed is predicate calculus with generalized quantifiers.

MRS was designed to be a tractable representation for large-scale parsing and generation, while not sacrificing expressiveness. It provides flat semantic representations that enable underspecification and can be integrated with grammatical representation in a number of frameworks. While MRS has been used in a wide variety of grammars, we concentrate here on MRS output by the English Resource Grammar (ERG, (Flickinger, 2000)), which we refer to as English Resource Semantics (ERS). The ERS is constructed compositionally in parallel with the syntactic analysis of a sentence.

To illustrate MRS/ERS, consider the sentence shown in (1) and the corresponding ERS in (2):[2]

(1) No generally accepted formal definition of algorithm exists yet.

(2) LTOP: l2,
RELS: < l4: _no(x, h7, _), l8: _general(e1, e2), l8: _accept(e2, _, x), l8: _formal(e3, x), l8: _definition(x, y), l5: udefq(y, h6, _), l3: _algorithm(y), l2: _exist(e4, x), l2: _yet(e5, e4), >
HCONS: < h7 $=_q$ l8, h6 $=_q$ l3 >

The main part of the representation is the RELS list, a bag of elementary predications (EPs). Each EP has an associated label, which is used for indicating scope (e.g., l8: _definition(x, y) has label l8). Predicates corresponding directly to word stems (i.e., lemmas) are indicated with a leading underscore. HCONS is a set of constraints which relate the labels to argument 'holes' in quantifiers and other scopal predicates. Local top (LTOP) is the topmost label in the MRS that is not the label of a quantifier. Note that there is a 'placeholder' quantifier, udefq, for the bare singular *algorithm*.[3] (3) shows the scoped readings in the object language:

(3) udefq(y, algorithm(y), _no(x, _general(e1, e2) ∧ _accept(e2, _, x) ∧ _formal(e3, x) ∧ _definition(x, y), _exist(e4, x) ∧ _yet(e5, e4)))
_no(x, udef(y, _algorithm(y), _general(e1, e2) ∧ _accept(e2, _, x) ∧ _formal(e3, x) ∧ _definition(x, y)) _exist(e4, x) ∧ _yet(e5, e4))

To show the relationship between these structures and MRS, first observe that through nesting of arguments each forms a tree, and that elements at each level of the tree are always combined with conjunctions. The root of the tree corresponds to the topmost quantifier. The MRS representation uses a

[2] From the 1214 version of the ERG: this is the top-ranked analysis produced, but the only other analysis is very similar. We have simplified the ERS in a number of respects for expository purposes.

[3] A kind term here, but the ERG does not distinguish kinds from ordinary entities since the syntax does not differentiate.

list of predications instead of the explicit conjunction. We can add an element to each predication to express its position in the tree: this is the MRS label. Instead of expressing the relationship between the quantifier (or other scopal predicate) and its arguments by embedding, we use a 'hole' argument to the quantifier and equate it to a label, as shown in (4):

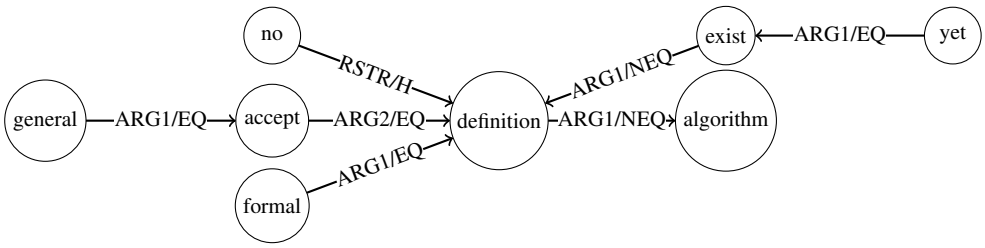
(4) l5: udefq(y, h6, l4), h6=l3, l4: _no(x, h7, l2), h7=l8, l8: _general(e1, e2), l8: _accept(e2, _, x), l8: _formal(e3, x), l8: _definition(x, y), l3: _algorithm(y), l2: _exist(e4, x), l2: _yet(e5, e4)

If we specify an LTOP and put the hole-label equalities into HCONS, this is now formally an MRS but it only corresponds to the first reading shown in (3). Scope underspecification in MRS is a generalization of the trees corresponding to the different scopes, maintaining the constraints between the elements via qeq constraints ($=_q$, equality modulo quantifier) between hole arguments and labels. Intuitively, a qeq constraint, $h =_q l$, enables one or more quantifiers to float between the label l and handle h but we will not explain the details here. Replacing the equalities with qeq constraints in the example above underspecifies scope, giving the MRS shown in (2).

Robust Minimal Recursion Semantics (RMRS) is a modified MRS representation that also allows underspecification of relational information (Copestake, 2007). The transformation process between MRS and RMRS splits off most of arguments of elementary predicates and refers to them using anchors (a). e.g., in (4), l8: _accept(e2, _, x) becomes l8:a4:_accept(e2), ARG2(a4, x).

Dependency MRS (DMRS) (Copestake, 2009) is an alternative representation interconvertible with MRS or RMRS. It has minimal redundancy in its structure and was developed for the purpose of readability and ease of use for both humans and computational applications. A DMRS is a directed graph with elementary predicates as nodes. It is constructed from a RMRS representation by combining 3 subgraphs: (1) Label equality graph, connecting EPs with shared labels; (2) Handle-to-label qeq graph, connecting handles and labels; (3) Variable graph, connecting EPs with their arguments. Upon merging the three subgraphs, the redundant links are deterministically removed to form a DMRS graph. The DMRS graph for our example is shown in (5):

(5)



Note that the udefq is missing in this DMRS, because we systematically ignore these placeholder predicates in the realization algorithm. For readability, we have not shown the leading underscores.

3 Hierarchical Statistical Semantic Realization

Hierarchical Statistical Semantic Realization (HSSR) is an approach that treats realization as a translation problem from a semantic representation to the surface string. As the input to realization is a DMRS graph, we define realization as transformation of a graph structure to a sequence of symbols. Transformation between the two is conducted using hierarchical rules, each rule realizing a part of the source graph. This approach draws inspiration from hierarchical phrase-based translation by Chiang (2005, 2007).

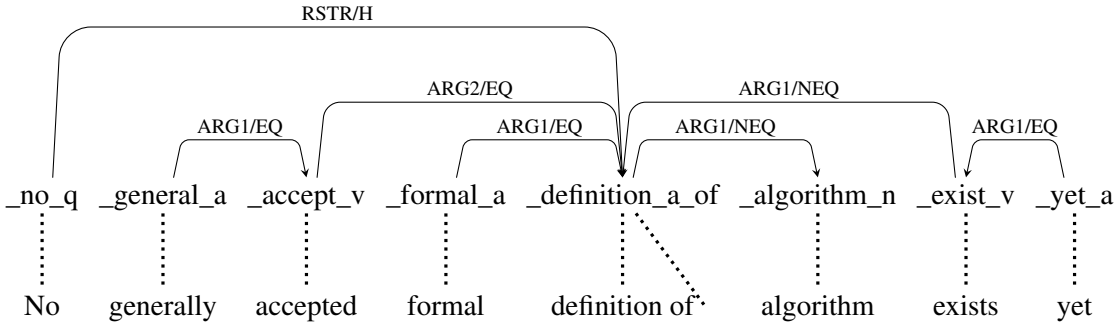
We refer to the collection of realization rules as a realization grammar. The realization grammar is automatically acquired from a large collection of MRS representations which are aligned with their sentence realizations.

We obtain a string realization of a previously unseen DMRS graph representation by applying a sequence of realization rules (a derivation) that transform all parts of the original DMRS graph. For any realization grammar of significant size, there will be many derivation candidates to choose from. We define a log-linear model over the derivations that assigns probabilities based on rule features and

N-gram language model probability. We obtain the final string realization by applying the most probable derivation to the source DMRS graph.

In the remainder of the section we describe the aspects of HSSR outlined above in more detail. Our description is accompanied by a realization example, whose source DMRS graph was shown in (5). In (6), the source graph is aligned with the string realization symbols:

(6)



3.1 Grammar

HSSR grammar is formally a Synchronous Context-Free Grammar (SCFG) consisting of rewrite rules of the form:

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle \tag{1}$$

where X is the nonterminal left-hand side, γ is a partial DMRS graph, α is a sequence of symbols, and $\sim$ is a one-to-one correspondence between nonterminal occurrences in γ and α. Consider the example rules extracted from (6):

(7) $X \rightarrow \langle$ _algorithm_n , algorithm $\rangle$

(8) $X \rightarrow \langle$ _definition_a_of _algorithm_n , definition of algorithm $\rangle$

ARG1/NEQ

(9) $X \rightarrow \langle$ _definition_a_of $X_{\boxed{1}}$, definition of $X_{\boxed{1}}$ $\rangle$

ARG1/NEQ

(10) $X \rightarrow \langle$ _no_q $X_{\boxed{1}}$ _exist_v _yet_a , no $X_{\boxed{1}}$ exists yet $\rangle$

RSTR/H ARG1/NEQ ARG1/EQ

We can interpret the rule in (8) as 'when encountering a subgraph consisting of two nodes, *_definition_a_of* and *_algorithm_n*, and an edge with label *ARG1/NEQ* originating in the former node and ending in the latter node, translate that subgraph as a sequence of symbols *definition of algorithm*.'

The rules shown in (7) and (8) consist of terminal nodes and terminal symbols. Additionally, a rule can contain one or more nonterminals X, represented as a nonterminal node on the source side and a nonterminal symbol on the target side. Two nonterminal rules extracted from (6) are shown in (9) and (10). The one-to-one correspondence $\sim$ between source and target side nonterminals is shown implicitly through the use of indexes on nonterminal symbols.

The presence of nonterminals in grammar rules enables hierarchical application of rules. Since every left hand side is the nonterminal X, any rule can be nested within any other rule with a nonterminal.

3.2 Rule extraction

HSSR rule extraction is an automatic procedure that extracts SCFG rules from a corpus to create a grammar. A grammar provides rules for translating previously unseen MRS representations. We define

the basic unit of rule extraction as an *example*, consisting of (1) a DMRS graph, (2) a sequence of symbols, and (3) the alignment between nodes and the symbols.

We can obtain such an example by parsing a sentence using a parser with a resource grammar. We used the ACE parser[4] combined with the English Resource Grammar. The parser produces an MRS representation, which we convert to a DMRS graph using the pyDelphin library[5]. Finally, we derive the alignment between nodes of the DMRS graph and symbols of the original sentence. Therefore, constructing a corpus suitable for rule extraction does not require any manual annotation. Instead, parsing a monolingual corpus with an MRS parser is sufficient.

The rule extraction procedure from a single example extracts terminal and nonterminal rules:

Terminal rules are extracted first for a given example and have the following properties:

1. The source side is a connected subgraph of the source graph consisting of terminal nodes.
2. The target side is a subsequence of terminal symbols.
3. No node outside the subgraph is aligned to a symbol in the target side and no symbol outside the target side is aligned to a node in the subgraph.
4. The rule contains at least one node aligned to at least one symbol.
5. The source side is a *valid* subgraph of the source graph. A valid subgraph contains a set of nodes such that no outgoing edge of any node is omitted. This ensures that only rules with all argument nodes present are extracted.

Nonterminal rules are extracted by subtracting a terminal rule from an existing rule. Rule subtraction replaces the subrule's subgraph and symbol sequence with a nonterminal symbol in the enclosing rule. Nonterminal rules have the following properties:

1. The source side is a connected subgraph of the source graph consisting of terminal and nonterminal nodes.
2. The target side is a sequence of terminal and nonterminal symbols.
3. No node outside the subgraph is aligned to a symbol in the target side and no symbol outside the target side is aligned to a node in the subgraph.
4. The rule contains at least one terminal node aligned to at least one terminal symbol.
5. Any pair of nonterminal nodes in the source subgraph does not share an edge.
6. A nonterminal node assumes no structure, i.e. the subtracted subgraph has a single node to which other unsubtracted nodes potentially connect to.
7. No edge originates from a nonterminal node.

(7) and (8) are terminal rules. The entire input graph shown in (6) can also form a terminal rule. (9) and (10) are examples of nonterminal rules. (9) was constructed by subtracting the terminal rule in (7) from the terminal rule in (8).

Nonterminal rules are extracted iteratively - in the first iteration, pairs of terminal rules are considered, while in the subsequent iterations, pairs of terminal rules and existing nonterminal rules are considered. This procedure produces rules with multiple nonterminals.

Manipulation of graphs, including enumerating subgraphs and comparing them, is inherently computationally intensive. To ensure that the rule extraction procedure is computationally tractable for sentences of reasonable length, we introduce two heuristic constraints on the rules in the final grammar: a) the size of the subgraph node set is at most five nodes; b) a rule contains at most two nonterminals.

Limiting the graph size to five nodes is a heuristic decision as we believe that five nodes are sufficient to capture most semantic locality in DMRS representations. We will verify this in future experiments. Limiting the number of nonterminals to two is a practical limitation to limit the size of extracted grammar and improve decoder efficiency.

[4]The ACE parser by Woodley Packard is available at http://sweaglesw.org/linguistics/ace/

[5]The pyDelphin library by Michael Goodman is available at https://github.com/goodmami/pydelphin

3.3 Model

A derivation is a sequence of translation rules that produces the full realization of an input representation. Any SCFG of significant size is able to produce many different derivations and consequently many different realizations of an input representation. A mechanism for choosing the best derivation is therefore needed. We define a log-linear model over derivations D:

$$P(D) \propto \prod_i \theta_i(D)^{\lambda_i} \tag{2}$$

where θ_i are features defined over rules used in derivation D and λ_i are feature weights. We define four features to aid realization: bidirectional conditional translation probabilities $P(\text{source}|\text{target})$ and $P(\text{target}|\text{source})$, N-gram language model probability, and word insertion penalty. The bidirectional probability features are trained by performing rule extraction and using rule frequency counts to estimate the probabilities. The feature weights λ_i of the log-linear model are tuned using grid search over the parameter space using BLEU (Papineni et al., 2002) as the measure of performance.

3.4 Decoder

The task of the decoder is to generate a string realization for a (previously unseen) MRS representation. The decoder uses a grammar estimated on a training corpus as the source of translation rules. We base the HSSR decoder on the ideas behind the HiFST hierarchical phrase-based translation system, presented in Iglesias et al. (2009).

Following the description in Allauzen et al. (2014), our decoder operates in three stages:

1. **Realization**: The decoder constructs a Weighted Finite State Acceptor (WFSA) encoding all possible realizations under a given synchronous context-free grammar G.

2. **Language Model application**: The decoder composes the realization WFSA with a weighted regular grammar defined by an N-gram language model. The resulting WFSA contains paths weighted by combined realization and language model scores.

3. **Search**: The decoder finds the shortest path through the combined WFSA.

We perform the **realization** stage in two distinct parts: (1) rule application, and (2) realization WFSA construction. Its implementation makes use of the OpenFST library[6] (Allauzen et al., 2007).

Given an input graph I and grammar G, our goal in rule application is to find the set of all rules R_I from grammar G, that can be used to realize graph I. Instead of checking all rules of grammar G against the graph I, we reverse the process. We generate all possible subgraphs I_s of the graph I that respect the same constraints we impose on the rules of grammar G in the rule extraction algorithm. Nevertheless, this process is less constrained than rule extraction due to the lack of a surface string. The set of subgraphs I_S forms a query against grammar G that retrieves the rules whose source side γ equals one of the subgraphs in the set I_S to form the set of all applicable rules R_I.

The query procedure requires matching of graphs against one another. This problem is commonly known as graph isomorphism problem, which belongs to the class of NP problems (Read and Corneil, 1977). We devised an efficient heuristic solution for DMRS graphs that works in the vast majority of cases. The heuristic solution fails only in the case of a completely symmetrical subgraph (in terms of node's adjacent and 1-removed neighbors), which is rarely encountered. When such a graph is encountered, the consequence is that some rules from the grammar that could have been applied to the subgraph are not recognized.

In an SMT environment using SCFGs a modified CYK algorithm operating over word spans is usually used to aid efficient construction of the translation WFSA. In contrast, instead of word spans HSSR decoder depends on the concept of *graph coverage*. A source side subgraph of a rule R covers a certain

[6] http://www.openfst.org

part of the input graph I. A graph coverage can be represented with a bit vector g_R of length n, where n is the size of the input graph's node set. Each position in g_R corresponds to a single node in the graph I. A particular position in g_R has a value of 1 if that node occurs in the subgraph of rule R, otherwise the value is 0. For instance, rule R in (9) has a node coverage bit vector $g_R = 00001100$, assuming that the order of nodes in the bit vector corresponds to the order they are displayed in (6). Graph coverage information is a byproduct of the rule application algorithm. Graph coverage with bit vectors resembles bit coverage of bag of words described in de Gispert et al. (2014).

While the size of the CYK grid of word spans grows in $\mathcal{O}(n^2)$ space with input size, the size of the grid of graph coverages grows in $\mathcal{O}(2^n)$. In general, however, the space of possible graph coverages of a given input graph is severely constrained by the rules present in the SCFG grammar.

In the second part of the realization stage, we use the set of applicable rules R_I and associated graph coverages g_R to create a WFSA which contains all possible realizations of the input graph I.

In a bottom-up process, the decoder groups rules into cells so that each cell corresponds to a single graph coverage bit vector. The decoder then encodes each R_i rule's target side as a Recursive Transition Network (RTN), treating each nonterminal arc as a pointer to a cell with a corresponding graph coverage g_R lower in the hierarchy. Based on rule features, a log-linear model assigns weights to arcs in RTNs.

A cell RTN is created as a union of all RTNs within that cell. Finally, the decoder performs a recursive RTN expansion, starting from the top most cell - the cell with the highest graph coverage (i.e. graph coverage with the largest number of nodes covered). RTN expansion replaces the nonterminal pointers to other cells lower in the hierarchy with the RTNs of those cells until reaching the bottom of hierarchy. RTN expansion produces the final realization WFSA encoding all realizations of the input graph I under grammar G.

The remaining two stages of the decoder are **language model application** and **search**. Language model application is conducted by composing the realization WFSA and language model WFSA using the *applylm* tool of the HiFST system[7] (Iglesias et al., 2009). The shortest path through the combined WFSA, found using the OpenFST shortest path operation, is the most probable realization under the given grammar and log-linear model, and therefore, the output of the decoder. In addition to finding a single shortest path through the WFSA, N-shortest paths can be extracted to produce an N-best list of realizations. An N-best list of realizations can be re-ranked using various strategies to improve the performance of the realizer. The strategies include re-ranking with a stronger language model and re-ranking using discriminative machine learning with a larger set of features.

4 Evaluation

We evaluated the HSSR approach by realizing a set of parsed MRS representations and comparing the realized surface strings against the original sentences. We use the BLEU metric for comparison of surface strings. Espinosa et al. (2010) have investigated the use of various automatic evaluation metrics to measure the quality of realization output. They have found that several standard Statistical Machine Translation evaluation metrics, including BLEU, correlate moderately well with human judgment of adequacy and fluency for the string realization task. The authors conclude that these metrics are useful for measuring incremental progress of a realization system, but advise caution when comparing different realization systems.

4.1 Experimental setup

We trained and evaluated the HSSR system on a subset of the Wikiwoods corpus. The Wikiwoods corpus, introduced by Flickinger et al. (2010), contains close to 1.3 million deep-parsed content articles, extracted from a snapshot of English Wikipedia in July 2008.

We randomly sampled chunks of the corpus to create our training, tuning, and test sets. The training set consists of around 1 million DMRS-sentence pairs. Tuning and test sets consist of 500 and 1000

[7]HiFST and related tools are available as open source: http://ucam-smt.github.io/

DMRS-sentence pairs respectively. Due to efficiency reasons, we selected input pairs of up to 20 tokens in the training set, and up to 15 DMRS graph nodes in the tuning and test sets. In future, we plan to address the efficiency of rule extraction and decoding with regards to input size by introducing an input graph splitting strategy, a graph equivalent to the standard practices of sentence splitting in SMT.

In the preprocessing stage, we performed general predicate node filtering from DMRS graphs to remove nodes that would introduce unnecessary complexity in rule extraction and decoding. On the other hand, we augmented the DMRS representations with explicit punctuation nodes and links, as the graphs otherwise do not contain information regarding punctuation.

We evaluated the system using 2-gram, 3-gram, and 4-gram language models. We estimated the language models on the entire Wikiwoods corpus, consisting of 800 million words (excluding tuning and training sets). The language models were estimated using KenLM toolkit (Heafield, 2011) with interpolated modified Kneser-Ney smoothing (Chen and Goodman, 1998).

Rule extraction on the training set of 1 million DMRS graph-surface string pairs produced 7.3 million realization rules. Practical limitations mentioned above apply: we extracted rules with at most 2 nonterminals, and the size of source side is at most five nodes.

We tuned the log-linear model weights using simple grid search over several iterations against the BLEU evaluation metric (Papineni et al., 2002). A mature system could instead be optimized using standard tuning approaches from SMT, for example MERT (Och, 2003) and LMERT (Macherey et al., 2008; Waite et al., 2011).

The decoding times of the current system implementation can be relatively long. We enforced reasonable computation time by terminating decoding of a DMRS graph after 300 seconds. This occurred for 96/1000 examples in the test set. As BLEU significantly penalizes short or omitted output using the brevity penalty, we computed the BLEU scores only on decoded examples. The final evaluation example set is the same between all systems in order to keep the scores comparable between them.

4.2 Results and discussion

We obtain the following results on decoded DMRS graphs of the test set:

Language model	BLEU
2-gram	46.02
3-gram	46.66
4-gram	46.63

We hypothesized that a strong language model will have a significant impact on the performance of the system. The hypothesis is partially confirmed by the increase in BLEU score from 2-gram to 3-gram. However, using a 4-gram language model yields no improvement over the 3-gram language model. The lack of improvement is unexpected, as 4-gram language models often yield good performance increases in SMT applications. The results warrant additional experiments with language models in future, including varying the size of training data for language model estimation and expanding the training data to other domains.

We list example output of the HSSR realization system in Table 1. In the first example, we show a successful instance of realization. The meaning is preserved by the realization process, although we could argue that '*grand falmouth hotel*' is not equivalent to '*grand hotel falmouth*'. The second example of realization is less successful, in part due to incorrect PP-attachment during parsing that produces a somewhat broken DMRS graph, and in part due to the application of incorrect realization rules.

As described in §3.4, the decoder produces the realization with the highest sum node coverage bit vector. In practical terms, this means that when it is not able to cover the entire source graph using hierarchical realization rules, it attempts to realize the largest part of the graph it can. This can be seen in the third realization example, where the substring '*, for instance ,*' was omitted from the realization as the grammar did not include a rule joining it with the rest of the sentence.

A more severe case of partial realization is shown in example four. The predicate *colour* has three arguments in the original DMRS representation. As we limit the number of nonterminals in rules to

	Source	Surface string
1.	original	the grand falmouth hotel was opened in 1865 just outside the station .
	realization	in 1865 the grand hotel falmouth was opened just outside the station .
2.	original	the railway would also open up areas of kazahstan for exploration of minerals and oil .
	realization	would also the railway to open up areas of kazahstan for exploration of minerals and oil .
3.	original	the direction also functioned as an inferior court in case of , for instance , theft .
	realization	the direction also functioned as an inferior court in case of theft .
4.	original	the false killer is uniformly coloured a dark grey to black .
	realization	a dark grey to black .
5.	original	although abuye survived this threat unharmed , sources differ on the details .
	realization	unharmed abuye survived this threat although sources differ on the details , .

Table 1: Output examples of the HSSR system with a 4-gram language model.

two, the only way to realize a predicate with three arguments is by using a rule that includes one of its arguments directly. Such a rule does not occur in the current grammar, and consequently the decoder is unable to realize the predicate *colour*. As the predicate node is the main verb of the sentence, it connects other parts of the the input graph together. For this reason, only the largest subgraph can be realized, producing '*a dark grey to black* .', despite the fact that a realization for '*the false killer*' is produced during decoding. We recognize three strategies of combating such errors: (1) increasing the maximum number of nonterminals to three, which would considerably increase the size of the realization grammar and the load on the decoder; (2) introducing glue rules which would combine realized parts of subgraphs despite missing connecting predicates; (3) increasing the size of the grammar in hope that such rules would be extracted from examples. We find the second preferable to others as it also enables realization in other instances of missing rules (not to mention that there are instances of predicates with four arguments). However, introduction of glue rules is nontrivial as it can create spurious ambiguity in the decoder - a situation where many distinct derivations with the same model features and realizations are produced (Chiang, 2007). An increase in spurious ambiguity would affect the decoder efficiency and cause problems for advanced tuning procedures depending on n-best lists, such as MERT.

The fifth and final example realization demonstrates the variability of output that the realization system is able to produce. This highlights the deficiencies of using N-gram precision measures such as BLEU for evaluating realization (and translation) output.

5 Related Work

In this paper we described a first attempt at statistical realization from MRS representations using synchronous context-free grammar. In this section we discuss similar approaches to realization.

One way of categorizing realization systems is according to the type of input assumed. Some authors have worked on the problem of word ordering, where the input is a bag of words, possibly combined with partial ordering information (e.g., Zhang and Clark (2011), de Gispert et al. (2014), Horvat and Byrne (2014)). Other systems take as input some form of meaning representation designed for a limited domain: such systems may be tested on the GEOQUERY corpus, for instance. Of particular relevance to us is Wong and Mooney (2007) which investigates SMT based techniques: they experiment with a phrase-based SMT method, a system that inverts an SMT-based semantic parser and a third approach which is a hybrid of these. Other systems take as input a structured representation which is intended to be flexible enough to cover general language: most such systems are associated with bidirectional approaches, and they have generally been tested with representations produced by parsers or with trees from manually-annotated treebanks. For instance, a number of systems have been built that take LFG f-structures as input (Cahill and van Genabith, 2006): others work on syntax trees, dependencies or logical representations.

These different assumptions about input make it extremely difficult to compare realization systems directly. The structural properties of the input determine which algorithms will be suitable for realization. While MRS has a logical object language, structurally the syntax of MRS is quite unlike a conventional

logic. The earlier work on MRS (i.e., Carroll et al. (1999) and subsequent papers listed in the introduction) used the flatness of its structure to facilitate the use of a chart generation approach, while in our work on generation from DMRS, the input is a graph.

In terms of methodology, our work is perhaps closest to (Cahill and van Genabith, 2006) whose PCFG system for robust probabilistic generation is based on approximations to a LFG automatically extracted from a treebank. However, the nature of the input and the techniques employed are very different. White (2011) investigates an approach which has some similarities with ours, using MT-style glue rules for robustness in conjunction with a realizer based on CCG, but his approach is directed at patching up failed realizations.

6 Conclusions and Future Work

In this paper, we presented a first attempt at statistical realization from MRS representations. The approach treats realization as a translation problem, transforming the Dependency MRS graph representation to a surface string. The HSSR approach draws inspiration from Statistical Machine Translation. We evaluated the performance of the new approach on a subset of the Wikiwoods corpus. We measured the performance of the HSSR system using BLEU and discussed its strengths and weakness using example output. The system shows promising results, with the main issues stemming from the lack of efficiency.

There are several areas of possible improvement. As mentioned above, the main area to address is decoder efficiency for larger input sizes and realization grammars. Additional heuristic constraints and WFSA pruning can be introduced to help decrease computational cost of realization. Realization of large DMRS graph representations (i.e. DMRS graphs with more than 20 nodes) may require a graph splitting strategy analogous to sentence splitting commonly performed in SMT. The performance of the realization system can likely be increased by extracting a larger realization grammar, engineering more features, and implementing an advanced tuning procedure such as MERT.

In addition to realization we are working to expand the HSSR system to an SMT system using semantic structures to aid the translation process. In this framework, DMRS graph representations in source language will be translated into surface strings in the target language.

References

Allauzen, C., B. Byrne, A. de Gispert, G. Iglesias, and M. Riley (2014). Pushdown Automata in Statistical Machine Translation. *Computational Linguistics 40*(3), 687–723.

Allauzen, C., M. Riley, J. Schalkwyk, W. Skut, and M. Mohri (2007). OpenFst : A General and Efficient Weighted Finite-State Transducer Library. In *Implementation and Application of Automata*, pp. 11–23.

Cahill, A. and J. van Genabith (2006, July). Robust PCFG-Based Generation Using Automatically Acquired LFG Approximations. In *Proceedings of COLING-ACL (2006)*, Sydney, Australia, pp. 1033–1040. Association for Computational Linguistics.

Carroll, J., A. Copestake, D. Flickinger, and V. Poznanski (1999). An efficient chart generator for (semi-) lexicalist grammars. In *European workshop on natural language generation*, pp. 86–95.

Carroll, J. and S. Oepen (2005). High Efficiency Realization for a Wide-Coverage Unification Grammar. In *IJCNLP*, pp. 165–176.

Chen, S. F. and J. T. Goodman (1998). An Empirical Study of Smoothing Techniques for Language Modeling. Technical report, Harvard University.

Chiang, D. (2005). A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of ACL 2005*, Michigan, USA, pp. 263–270.

Chiang, D. (2007). Hierarchical phrase-based translation. *Computational Linguistics 33*(2), 201–228.

Copestake, A. (2007). Semantic Composition with (Robust) Minimal Recursion Semantics. In *ACL 2007 Workshop on Deep Linguistic Processing*, pp. 73–80.

Copestake, A. (2009). Slacker semantics: why superficiality, dependency and avoidance of commitment can be the right way to go. In *Proceedings of EACL 2009*, Athens, Greece, pp. 1–9.

Copestake, A., D. Flickinger, R. Malouf, S. Riehemann, and I. Sag (1995). Translation using Minimal Recursion Semantics. In *Proceedings of the 6th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI-95)*, Leuven, Belgium.

Copestake, A., D. Flickinger, C. Pollard, and I. Sag (2005, July). Minimal Recursion Semantics: An Introduction. *Journal of Research on Language and Computation 3*(2-3), 281–332.

de Gispert, A., M. Tomalin, and W. Byrne (2014). Word Ordering with Phrase-Based Grammars. In *Proceedings of EACL 2014*, pp. 259–268.

Espinosa, D., R. Rajkumar, M. White, and S. Berleant (2010). Further Meta-Evaluation of Broad-Coverage Surface Realization. In *Proceedings of EMNLP 2010*, pp. 564–574.

Flickinger, D. (2000). On building a more effcient grammar by exploiting types. *Natural Language Engineering 6*(1), 15–28.

Flickinger, D., S. Oepen, and G. Ytrestø l (2010). WikiWoods: Syntacto-Semantic Annotation for English Wikipedia. In *Proceedings of LREC 2010*, Valletta, Malta, pp. 1665–1671.

Heafield, K. (2011). KenLM : Faster and Smaller Language Model Queries. In *Proceedings of the EMNLP 2011 Sixth Workshop on Statistical Machine Translation*, Edinburgh, UK, pp. 187–197.

Horvat, M. and W. Byrne (2014). A Graph-Based Approach to String Regeneration. In *Student Research Workshop at EACL 2014*, Gothenburg, Sweden, pp. 85–95.

Iglesias, G., A. de Gispert, E. R. Banga, and W. Byrne (2009). Hierarchical Phrase-Based Translation with Weighted Finite State Transducers. In *Proceedings of HLT-NAACL 2009*, Boulder, USA, pp. 433–441.

Lø nning, J. T., S. Oepen, D. Beermann, L. Hellan, J. Carroll, H. Dyvik, D. Flickinger, J. B. Johannessen, P. Meurer, T. r. Nordgard, V. Rosen, and E. Velldal (2004). LOGON - A Norwegian MT Effort. In *Proceedings of the Workshop in Recent Advances in Scandinavian Machine Translation*, Uppsala, Sweden.

Macherey, W., F. J. Och, I. Thayer, and J. Uszkoreit (2008). Lattice-based minimum error rate training for statistical machine translation. In *Proceedings of EMNLP 2008*, Honolulu, USA, pp. 725–734.

Och, F. J. (2003). Minimum error rate training in statistical machine translation. In *Proceedings of ACL 2003*, pp. 160–167.

Papineni, K., S. Roukos, T. Ward, and W.-j. Zhu (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of ACL 2002*, Philadelphia, USA, pp. 311–318.

Read, R. C. and D. G. Corneil (1977). The graph isomorphism disease. *Journal of Graph Theory 1*(4), 339–363.

Velldal, E. (2009). *Empirical Realization Ranking*. Ph. D. thesis, University of Oslo.

Velldal, E. and S. Oepen (2005). Maximum Entropy Models for Realization Ranking. In *Machine Translation Summit*, pp. 109–116.

Waite, A., G. Blackwood, and W. Byrne (2011). Lattice-based minimum error rate training using weighted finite-state transducers with tropical polynomial weights. In *Proceedings of the 9th International Workshop on Finite-State Methods and Natural Language Processing (FSMNLP 2011)*, Blois, France, pp. 116–125.

White, M. (2011). Glue rules for robust chart realization. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pp. 194–199.

Wong, Y. W. and R. J. Mooney (2007). Generation by Inverting a Semantic Parser that Uses Statistical Machine Translation. In *HLT-NAACL*, pp. 172–179.

Zhang, Y. and S. Clark (2011). Syntax-Based Grammaticality Improvement using CCG and Guided Search. In *Proceedings of EMNLP 2011*, pp. 1147–1157.