The Alexa Meaning Representation Language

Thomas Kollar, Danielle Berry, Lauren Stuart, Karolina Owczarzak, Tagyoung Chung
{kollart, danijean, lsstuar, karowc, tagyoung}@amazon.com

Lambert Mathias, Michael Kayser, Bradford Snow, Spyros Matsoukas
{mathiasl, mikayser, brsnow, matsouka}@amazon.com

Abstract

This paper introduces a meaning representation for spoken language understanding. The Alexa meaning representation language (AMRL), unlike previous approaches, which factor spoken utterances into domains, provides a common representation for how people communicate in spoken language. AMRL is a rooted graph, links to a large-scale ontology, supports cross-domain queries, fine-grained types, complex utterances and composition. A spoken language dataset has been collected for Alexa, which contains $\sim$ 20k examples across eight domains. A version of this meaning representation was released to developers at a trade show in 2016.

1 Introduction

Amazon has developed Alexa, a voice assistant that has been deployed across millions of devices and processes voice requests in multiple languages. This paper addresses improvements to the Alexa voice service, whose core capabilities (as measured by the number of supported intents and slots) has expanded more than four-fold over the last two years. In addition more than ten thousand voice skills have been created by third-party developers using the Alexa Skills Kit (ASK). In order to continue this expansion, new voice experiences must be both accurate and capable of supporting complex interactions.

However, as the number of features has expanded, adding new features has become increasingly difficult for four primary reasons. First, requests with a similar surface form may belong to different domains, which makes it challenging to add features without degrading the accuracy of existing domains. For example, similar linguistic phrases such as *"order me an echo dot"* (e.g., for Shopping) have a similar form to phrases used for a ride-hailing feature such as, *"Alexa, order me a taxi"*. The second challenge is that a fixed flat structure is unable to easily support certain features (Gupta et al., 2006b), such as cross-domain queries or complex utterances, which cannot be clearly categorized into a given domain. For example, *"Find me a restaurant near the sharks game"* contains both local businesses and sporting events and *"Play hunger games and turn the lights down to 3"* requires a representation that supports assigning an utterance to two intents. The third challenge is that there is no mechanism to represent ambiguity, forcing the choice of a fixed interpretation for ambiguous utterances. For example, *"Play Hunger Games"* could refer to an audiobook, a movie, or a soundtrack. Finally, representations are not reused between skills, leading to the need for each developer to create a custom data and representations for their voice experiences.

In order to address these challenges and make Alexa more capable and accurate, we have developed two key components. The first is the Alexa ontology, a large hierarchical ontology that contains fine-grained types, properties, actions and roles. Actions represent a predicate that determines what the agent should do, roles express the arguments to an action, types categorize textual mentions and properties are relations between type mentions. The second component is the Alexa Meaning Representation Language (AMRL), a graph-based domain and language independent meaning representation that can capture the meaning of spoken language utterances to intelligent assistants. AMRL is a rooted graph where action, operators, relations and classes are labeled vertices and properties and roles are labeled edges. Unlike typical representations for spoken language understanding (SLU), which factors language understanding into the prediction of intents (non-overlapping actions) and slots (e.g., named entities) (Gupta et al., 2006a), our representation is grounded in the Alexa ontology, which provides a common semantic representation for spoken language understanding and can directly represent ambiguity, complex nested utterances and cross-domain queries. Unlike similar meaning representations such as AMR (Banarescu et al., 2013), AMRL is designed to be cross-lingual, explicitly represent fine-grained entity types, logical statements, spatial prepositions and relationships and support type mentions. Examples of AMRL and the SLU representations can be seen in Figure 1.

The AMRL has been released via Alexa Skills Kit (ASK) built-in intents and slots in 2016 at a developers conference, offering coverage for eight of the ∼20 SLU domains [1]. In addition to these domains, we have demonstrated that the AMRL can cover a wide range of additional utterances by annotating a sample from all first and third-party applications. We have manually annotated data for 20k examples using the Alexa ontology. This data includes the annotation of ∼100 actions, ∼500 types, ∼20 roles and ∼172 properties.

2 Approach

This paper describes a common representation for SLU, consisting of two primary components:

- The Alexa ontology - A large-scale hierarchical ontology developed to cover all spoken language usage.
- The Alexa meaning representation language (AMRL) - A rooted graph that provides a common semantic representation, is compositional and can support complex user requests.

These two components are described in the following sections.

2.1 The Alexa ontology

The Alexa ontology provides a common semantics for SLU. The Alexa ontology is developed in RDF and consists of five primary components:

- **Classes** A hierarchy of Classes, also referred to as types, is defined in the ontology. This hierarchy is a rooted tree, with finer-grained types at deeper levels. Coarse types that are children of THING include PERSON, PLACE, INTANGIBLE, ACTION, PRODUCT, CREATIVEWORK, EVENT and ORGANIZATION. Fine-grained types include MUSICRECORDING and RESTAURANT.
- **Properties** A given class contains a list of properties, which relate that class to other classes. Properties are defined in a hierarchy, with finer-grained classes inheriting the properties of its parent. There are range restrictions on the available types for both the domain and range of the property.
- **Actions** A hierarchy of actions are defined as classes within the ontology. ACTIONS cover the core functionality of Alexa.
- **Roles** ACTIONS operate on entities via roles. The most common role for an ACTION is the **.object** role, which is defined to be the entity on which the ACTION operates.
- **Operators and Relations** A hierarchy of operators and relations represent complex relationships that cannot be expressed easily as properties. Represented as classes, these include ComparativeOperator, Equals and Coordinator (Figure 2).

The Alexa ontology utilized schema.org as its base and has been updated to include support for spoken language. In addition, using schema.org as the base of the Alexa Ontology means that it shares a vocabulary used by more than 10 million websites, which can be linked to the Alexa ontology.

2.2 Alexa meaning representation language

AMRL leverages classes, properties, actions, roles and operators in the main ontology to create a compositional, graph-based representation of the meaning of an utterance. The graph-based representation conceptualizes each arc as a property and each node as an instance of a type; each type can have multiple parents. Conventions have been developed to annotate the AMRL for an utterance accurately and consistently. These conventions focus primarily on linguistic annotation, and only consider filled pauses, edits, and repairs in limited contexts. The conventions include:

- **Fine-grained type mentions** When an entity type appears in an utterance, the most fine-grained type will be annotated. For *"turn on the light"*, the mention *'light'* could be annotated as a DEVICE. However, there is a more appropriate finer-grained type, LIGHTING which will be selected instead.
- **Ambiguous type mentions** When more than one fine-grained type is possible, then the annotator will utilize a more coarse-grained

[1]https://amzn.to/2qDjNcJ

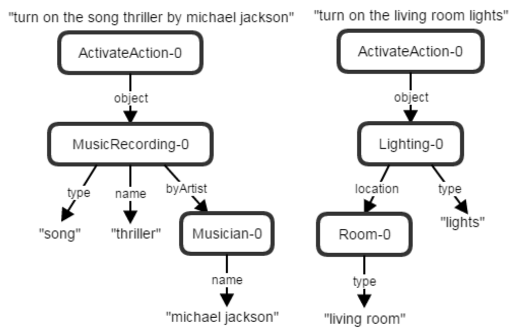
"turn on the song thriller by michael jackson"

D	MusicApp
I	ListenMediaIntent
S	turn on the [song]$_{\text{SongType}}$ [thriller]$_{\text{Song}}$ by [michael jackson]$_{\text{Singer}}$

"turn on the living room lights"

D	HomeAutomation
I	ActivateIntent
S	turn on the [living room]$_{\text{Location}}$ [lights]$_{\text{Device}}$

(a) SLU Example 1



(b) AMRL Example 2

Figure 1: This figure shows the SLU representation on the left and the AMRL representation on the right. The three components of the SLU representation, domain (D), intent (I) and slots (S) are shown. The intent is different (e.g., "ListenMediaIntent" vs. "ActivateIntent"), despite the presence of*"turn on"*. On the right are the same utterances represented in the AMRL. The nodes represent the instances of classes defined in an ontology, while the directed arcs connecting the class instances are properties. The root node of both graphs is the action, ACTIVATEACTION is shared across these two utterances, providing the domain-less annotation with a uniform representation for the same carrier phrase. "-0" indicates the first mention of a type in the utterance, and can be used used to denote co-reference across multiple dialog turns.

type in the hierarchy. This type should be the finest-grained type that still captures the ambiguity. For example, in the utterance *"play thriller'*, "thriller" can either be a MUSICALBUM or a MUSICRECORDING. Instead of selecting one of these a more coarse-grained type of MUSICCREATIVEWORK will be chosen. When the ambiguity would force fallback to the root class of the ontology THING, AMRL annotation chooses a sub-class and marks the usage of it as *uncertain*.

- **Properties** Properties are annotated when they are unambiguous. For example, *"find books by truman capote"*, the use of the **.author** property on the BOOK class is unambiguous. Similarly, for *"find books about truman capote"* the use of the **.about** property on the BOOK class is unambiguous.
- **Ambiguous property usage** When there is uncertainty in the property that should be selected for the representation, the annotator may fall back to a more generic property.
- **Property inverses** When a property can be annotated in two different directions, a canonical property is defined in the ontology and used for all annotations. For example, **.parentOrganization** has an inverse of **.subOrganization**. The former is selected as canonical for annotation flexibility and to eliminate cycles in the graph.

A few of these properties have special meaning at annotation time. Specifically, for the annotation of textual mentions there exist three primary properties: **.name**, **.value** and **.type**. The conventions for these properties are as follows:

- **.name** This is a nominal mention in the utterance, the **.name** property links the text to an instance of a class. **.name** is only used for mentions that are not a numeric quantity or enumeration. An example of **.name** for a MUSICIAN class would be *"madonna"*.
- **.value** This is defined in the same way as **.name** but is used for mentions that are numeric quantities or enumerations. For instance, *"two"* would be a **.value** of an INTEGER class.
- **.type** This is a generic mention of an entity type. For example, *"musician"* is a **.type** mention of the MUSICIAN class.

One action (NULLACTION) has a special meaning. This is annotated whenever a SLU query does not have an associated action or the action is unclear. This happens, for example, when someone says, *"temperature"*. In contrast, *"show me the temperature"* is annotated with the more specific DISPLAYACTION.

2.3 Expanded Language Support

AMRL has been used to represent utterances that are either not supported or challenging to support using standard SLU representations. The following section describes support for anaphora, complex and cross-domain utterances, referring expressions for locations and composition.

2.3.1 Anaphora

AMRL can natively support pronominal anaphora resolution both within the same utterance or across utterances. For example:

- Within utterance: *"Find the highest-rated toaster and show me its reviews"*
- Across utterances: *"What is Madonna's latest album" "Play it."*

Terminal nodes refer back to the same (unique) entity. An example annotation across multiple utterances can be seen in Figures 3a and 3b. Similar to the above, it can handle bridges within discourse, such as, "find me an italian restaurant" and "what's on its menu."

2.3.2 Inferred nodes

AMRL contains nodes that are not grounded in the text. For example, for the utterance, in Figure 2a there are two inferred nodes, one for the address of the restaurant and another for the address of the sports event. Not explicitly representing types has two primary benefits. First, certain linguistic phenomena such as anaphora are easier to support. Second, the representation is aligned to the ontology, which enables direct queries against the knowledge base. Inferred nodes are the AMRL way to perform reification.

2.3.3 Cross-domain utterances

Using the common semantics of AMRL means that parses do not need to obey domain boundaries. For example, these utterances would belong to two domains (e.g., sports and local search): *"Where is the nearest restaurant"* and *"What is happening at the Sharks game"*. AMRL, as in Figure 2a, can handle utterances that span multiple domains, such as the one shown in Figure 2a.

2.3.4 Conjunctions, disjunctions and negations

AMRL can cover logical expressions, where there can be an arbitrary combination of conjunctions, disjunctions, or conditional statements. Some examples of object-level or clause-level conjunctions include:

- Object-level conjunction: *"Add milk, bread, and eggs to my shopping list"*
- Clause-level conjunction: *"Restart this song and turn the volume up to seven"*

Conjunctions and disjunctions are represented using a Coordinator class. The ".value" property defined which logical operation is to be performed. Examples of the AMRL representation for these is shown in Figure 2b and 2c.

2.3.5 Conditional statements

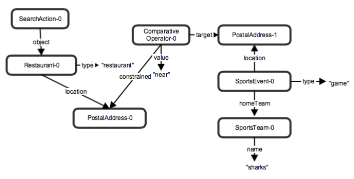
Conditional statements are not usually represented in other formalisms. An example of a conditional statement is, *"when its raining, turn off the sprinklers"*. Time-based conditional statements are special cased due to their frequency in spoken language. For time-based expressions (e.g., *"when it is three p.m., turn on the lights"*), a startTime (or endTime) property is used on the action to denote the condition of when the action should start (or stop). For all other expressions, we use the ConditionalOperator, which has a "condition" property as well as a "result" property. When the condition is true, then the result would apply. The constrained properties are defining the arguments of the Equals operator. An example can be seen in Figure 4. A deterministic transformation from the simplified time-based scheme to ConditionalOperator form when greater consistency is desired.

2.3.6 Referring expressions for locations

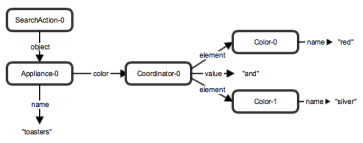
AMRL can represent locations and their relationships. For simpler expressions that are common, such as "on" or "in," properties are used to represent the relationship between two entity mentions. For other spatial relations, such as "between" or "around," an operator is introduced. Two examples of spatial relationships can be seen in Figure 2d. In this example "beside" grounds to the relation being used (e.g., "beside") and uses two properties (e.g., constrained and target), which are the the first and second arguments to the spatial preposition.

2.3.7 Composition

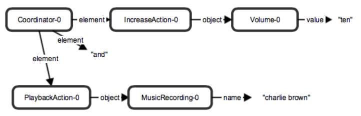
AMRL supports composition, which enables reuse of types and subgraphs to represent utterances with similar meanings. For example, Figures 2e and 2f show the ability to create significantly different actions only by changing the type of the object of the utterance. Such substitution can occur



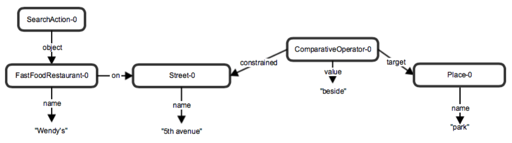
(a) Cross-domain



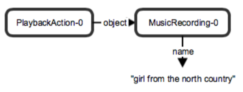
(b) Conjunction



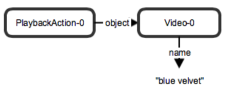
(c) Conjunction (multiple actions)



(d) Spatial relations

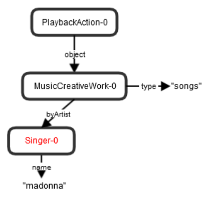


(e) Composition (1)

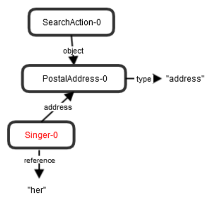


(f) Composition (2)

Figure 2: Examples of complex queries. In (a) is the utterance *"find restaurants near the sharks game."*. In (b) is the utterance *"find red and silver toasters"*. In (c) is *"play charlie brown and turn the volume up to 10"*. In (d) is *"find the wendy's on 5th avenue beside the park."* In (e) and (f) are an illustration composition for, *"play girl from the north country"* and *"play blue velvet."*.



(a) Turn 1



(b) Turn 2

Figure 3: (a) shows the first turn of a conversation, *"play songs by madonna"* (b) shows the second turn of a conversation, *"what's her address"*. Because the node SINGER-0 has the same "-0" ID in both turns, the previous turn can be directly used to infer that the address should be for the person whose name is "Madonna."

anywhere in the annotation graph. PlaybackAction is used to denote playing of the entity referred to by the object role.

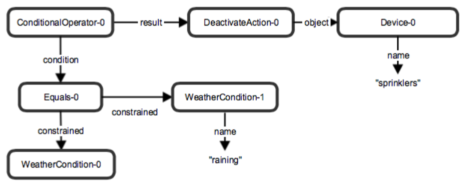
2.3.8 Unsupported features

Although many linguistic phenomena can be supported in AMRL, there are a few that have not been explicitly supported and are left for future work. These include existential and universal quantification and scoping and conventions for agency (most requests are imperative). In addition, there is currently no easy way to convert to first order logic (e.g., lambda calculus), due to conventions that simplify annotation, but lose information about operators such as spatial relationships.

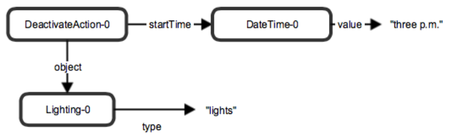
3 Dataset

Data has been collected for the AMRL across many spoken language use-cases. The current domains that are supported include music, books, video, local search, weather and calendar. We have prototyped mechanisms to speed up annotation via paraphrasing (Berant and Liang, 2014) and conversion from our current SLU representation, in order to leverage the much larger data available. The primary mechanism we have for data-acquisition is via manual annotation. Tools have been developed in order to acquire the full graph annotated with all the properties, classes, actions and operators.

AMRL manual annotation is performed by data annotators in four stages. In the first stage an action is selected, for example ACTIVATEACTION in Figure 1b. The second stage defines the text spans in an utterance that link to a class in the ontology (e.g., "michael jackson" is a Musician type and "thriller" and "song" are MusicRecording types, the first is a **.name** mention, while the latter is a **.type** mention. The third stage creates connections between the classes and defines any missing nodes in the graph. In the final stage a skilled annotator reviews the graph for mistakes and and re-annotates it if necessary. There is a visualization of the semantic annotation available, enabling an annotator to verify that they have built the graph in a semantically accurate manner. Manual annotation happens at the rate of 40 per hour. The manually annotated dataset contains ∼20k annotated utterances and contains 93 unique actions,



(a) AMRL for "when it is raining, turn off the sprinklers"



(b) AMRL for "when it is three p.m., turn on the lights."

Figure 4: Two examples of conditional statements. In (b) are the annotation for time-based conditions, while in (a) is a non-time based trigger.

448 types, 172 properties and 23 roles.

4 Parsing

Any graph parsing method can be used to predict AMRL given a natural language utterance. One approach is to use hyperedge replacement grammars (Chiang et al., 2013) (Peng et al., 2015), though these require large datasets in order to train accurate parsers. Alternatively, the graph can be linearized, as in (Gildea et al., 2017) and sequence to sequence or sequential models can be used to predict AMRL (Perera and Strubell, 2018). We have shown that AMRL full-parse accuracy is at 78%, though the serialization, use of embeddings from related tasks can improve parser accuracy. More details can be found in (Perera and Strubell, 2018).

5 Related Work

FreeBase (Bollacker et al., 2008) (now WikiData) and schema.org (Guha et al., 2016) are two common ontologies. Schema.org is widely used on the web and contains actions, types and properties. The Alexa ontology expands schema.org to cover types, properties and roles used in spoken language.

Semantic parsing has been investigated in the content of small domain-specific datasets such as GeoQuery (Wong and Mooney, 2006) and in the context of larger broad-coverage representations such as the Groningen Meaning Bank (GMB) (Bos et al., 2017), the Abstract Meaning Representation (AMR) (Banarescu et al., 2013), UCCA (Abend and Rappoport, 2013), PropBank (Kingsbury and Palmer, 2002), Raiment (Baker et al., 1998) and lambda-DCS (Kingsbury and Palmer, 2002). OntoNotes (Hovy et al., 2006), lambda-DCS s (Liang, 2013) (Baker et al., 1998), FrameNet (Baker et al., 1998), combinatory categorial grammars (CCG) (Steedman and Baldridge, 2011) (Hockenmaier and Steedman, 2007), universal dependencies (Nivre et al., 2016) are all related representations. A comparison of semantic representations for natural language semantics is described in Abend and Rappoport. Unlike these meaning representations for written language, AMRL covers question answering, imperative actions, and a wide range of new types and properties (e.g., smart home, timers, etc.).

AMR and AMRL are both rooted, directed, leaf-labeled and edge-labeled graphs. AMRL does not reuse PropBank frame arguments, covers predicate-argument relations, including a wide variety of semantic roles, modifiers, co-reference, named entities and time expressions (Banarescu et al., 2013). There are more than 1000 named-entity types in AMRL (AMR has around 80). Reentrancy is not used in AMRL notation. In addition to the AMR "name" property, AMRL contains a "type" property for mentions of a type (or class) and a "value" property for the mention of numeric values. Anaphora is handled in AMRL for spoken dialog Poesio and Artstein (Gross et al., 1993). Unlike representations used for spoken language understanding (SLU) (Gupta et al., 2006b), AMRL represents both entity spans, complex natural language expressions, and fine-grained named-entity types.

6 Conclusions and Future Work

This paper develops AMRL, a meaning representation for spoken language. We have shown how it can be used to expand the set of supported use-cases to complex and cross-domain utterances, while leveraging a single compositional semantics. The representation has been released at AWS Re:Invent 2016 [2]. It is also being used as a representation for expanded support for complex utterances, such as those with sequential composi-

[2]https://amzn.to/2qDjNcJ

tion. Continued development of a common meaning representation for spoken language will enable Alexa to become capable and accurate, expanding the set of functionality for all Alexa users.

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *ACL (1)*. pages 228–238.

Omri Abend and Ari Rappoport. 2017. The state of the art in semantic representation. In *Proceedings of the Association for Computational Linguistics*. Vancouver,CA.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '98, pages 86–90. `https://doi.org/10.3115/980845.980860`.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*. Association for Computational Linguistics, Sofia, Bulgaria, pages 178–186. `http://www.aclweb.org/anthology/W13-2322`.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*. ACM, New York, NY, USA, SIGMOD '08, pages 1247–1250. `https://doi.org/10.1145/1376616.1376746`.

Johan Bos, Valerio Basile, Kilian Evang, Noortje Venhuizen, and Johannes Bjerva. 2017. The groningen meaning bank. In Nancy Ide and James Pustejovsky, editors, *Handbook of Linguistic Annotation*, Springer, volume 2, pages 463–496.

David Chiang, Jacob Andreas, Daniel Bauer, Karl Moritz Hermann, Bevan Jones, and Kevin Knight. 2013. Parsing graphs with hyperedge replacement grammars. In *ACL (1)*. pages 924–932.

Daniel Gildea, Nianwen Xue, Xiaochang Peng, and Chuan Wang. 2017. Addressing the data sparsity issue in neural amr parsing. In *EACL*.

Derek Gross, James Allen, and David Traum. 1993. The trains 91 dialogues .

R. V. Guha, Dan Brickley, and Steve Macbeth. 2016. Schema.org: Evolution of structured data on the web. *Commun. ACM* 59(2):44–51. `https://doi.org/10.1145/2844544`.

N. Gupta, G. Tur, D. Hakkani-Tur, S. Bangalore, G. Riccardi, and M. Gilbert. 2006a. The at&t spoken language understanding system. *IEEE Transactions on Audio, Speech, and Language Processing* 14(1):213–222. `https://doi.org/10.1109/TSA.2005.854085`.

Narendra Gupta, Gokhan Tur, Dilek Hakkani-Tur, Srinivas Bangalore, Giuseppe Riccardi, and Mazin Gilbert. 2006b. The AT&T spoken language understanding system. *Audio, Speech, and Language Processing, IEEE Transactions on* 14(1):213–222.

Julia Hockenmaier and Mark Steedman. 2007. Ccgbank: A corpus of ccg derivations and dependency structures extracted from the penn treebank. *Comput. Linguist.* 33(3):355–396. `https://doi.org/10.1162/coli.2007.33.3.355`.

Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. 2006. Ontonotes: The 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*. Association for Computational Linguistics, Stroudsburg, PA, USA, NAACL-Short '06, pages 57–60. `http://dl.acm.org/citation.cfm?id=1614049.1614064`.

Paul Kingsbury and Martha Palmer. 2002. From treebank to propbank. In *LREC*. pages 1989–1993.

Percy Liang. 2013. Lambda dependency-based compositional semantics. *CoRR* abs/1309.4408. `http://arxiv.org/abs/1309.4408`.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal dependencies v1: A multilingual treebank collection. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Sara Goggi, Marko Grobelnik, Bente Maegaard, Joseph Mariani, Helene Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*. European Language Resources Association (ELRA), Paris, France.

Xiaochang Peng, Linfeng Song, and Daniel Gildea. 2015. A synchronous hyperedge replacement grammar based approach for amr parsing. In *CoNLL*. pages 32–41.

Vittorio Chung Tagyoung Kollar Thomas Perera and Emma Strubell. 2018. Multi-task learning for parsing the alexa meaning representation language. In *American Association for Artificial Intelligence (AAAI)*.

Massimo Poesio and Ron Artstein. 2008. Anaphoric annotation in the arrau corpus. In *LREC*.

Mark Steedman and Jason Baldridge. 2011. *Combinatory Categorial Grammar*, Wiley-Blackwell, pages 181–224. `https://doi.org/10.1002/9781444395037.ch5`.

Yuk Wah Wong and Raymond J Mooney. 2006. Learning for semantic parsing with statistical machine translation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*. Association for Computational Linguistics, pages 439–446.