Spreading activation in language understanding

Dávid Nemeskey, Gábor Recski, Márton Makrai, Attila Zséder, András Kornai
HAS Computer and Automation Research Institute
H-1111 Kende u 13-17, Budapest
{ndavid,recski,makrai,zseder,kornai}@sztaki.mta.hu

ABSTRACT

We describe our implementation of a natural language understanding capable of acting on fragmentary input. The key building blocks of our system are generalized finite state automata which contain both syntactic and semantic information about the words and larger constructions, regulating the interaction of these with external knowledge sources.

1. INTRODUCTION

This paper describes a simple word-driven architecture for a dialog system capable of acting on the kind of fragmentary input that pervades naturally occurring dialog between humans in service encounters. Our system follows in the tracks of widely deployed finite state analyzers such as SRI's FASTUS (Appelt 1993) and the Xerox chunkers (Grefenstette 1999), but integrates the syntactic and semantic subsystems so tightly that there is no need for a separate control structure interleaving the two.

While the ATIS corpus (Hemphill et al 1990) was already collected in a Wizard-of-Oz paradigm to encourage natural interaction, here we went a step further and recorded actual requests made to a human ticket clerk at a railroad station. The resulting Hungarian Railroad Co. (MÁV) corpus is dominated by fragments such as '**please** give me a ticket from here **to Göd** with **a pensioner** discount' wherein only the bolded parts were actually produced, the rest had to be inferred. This task apparently caused no difficulty to the person behind the counter: over 90% of the requests were fulfilled without any need for clarification.

We use an asynchronous communication system with three component types. *Plugins* provide interfaces to the services that the system needs to access. These objects implement an abstraction layer that enables handling a web search engine, such as that of Wikipedia, in the same way as a NLP tool on the local machine. Plugins work on a request-response basis, and are by default stateless. *Agents* represent autonomous components in the system – they are similar to plugins, except that they can initiate communication by themselves. Each user is represented by an agent object, which provides a text-based conversation interface to the system. Other active external components, such as cameras, could also be wrapped into agents. Finally, there are the syntacto-semantic components, called *machines*, which correspond both to stored morpheme- and word-sized lexical units and to dynamically built larger (phrase- or sentence-sized) units of analysis.

Section 2 describes machines, a generalization of finite-state automata (FSA) and finite-state transducers (FSTs) due to Eilenberg (1974). Section 3 describes how linguistic and nonlinguistic inputs are processed, and how the resulting activation pattern spreads through the system to produce useful responses. Section 4 evaluates the work on the MÁV and ATIS corpora. Some conclusions are offered in Section 5.

2. MACHINES, AVMS, AND THE LEXICON

Machines are composed of an FSA over some alphabet Σ, a base set X, and a mapping from Σ to the set of binary relations over X. They were introduced by Eilenberg (1974) as a means of formalizing the flowcharts that were commonly used at the time for depicting algorithms. As the FSA consumes a letter from some string over Σ, it moves, possibly nondeterministically, into some other state. The transition will have some side effect, mapping elements of X, again possibly nondeterministically (hence the use of relations rather than functions) onto themselves, depending only on the letter consumed, not on the state of the FSA before or after the transition. While the state and arc sets of FSA are by definition finite, the base set X of a machine may be infinite.

Attribute-value matrices (AVMs) are standard data structures that lack the kind of dynamic behavior that FSA, FST, and machines have. They are stateful only in the sense of being more or less filled. Both in relation extraction (database population) and in dialog management the overarching goal of the analysis is to find the relevant values for each attribute. We cannot consider the ticket clerk task completed until we know the values of the ORIGIN, DESTINATION, TIME_RANGE and FARE_CLASS attributes, but once these values are known, the ticket can be issued by a printer plugin that embodies no natural language understanding whatsoever. (Receiving a sufficient amount of money may be considered a precondition of issuing a ticket, but we shall ignore details of the debit process here, as it takes place outside the natural language component.)

Kornai (2010) recognized that machines can be used to keep track of the partially filled state of the case frame of a verb, a task standardly performed by a computationally much stronger mechanism, variable binding. Here we extend the idea to AVMs: simultaneous with the FSA consuming a case-marked NP such as the sublative **to Göd**, we should add the NP as a value to the

DESTINATION attribute. The FSA can be used to exclude this slot from the set of still-unfilled attributes, and the relational composition mechanism can do the actual slot-filling. Using the terminology of Curry (1961), the FSA component of a machine is responsible for the phenogrammar and the relations over X coupled to the alphabet express the tectogrammar.

In the early stages, the analysis proceeds in the standard fashion: words undergo morphological analysis and NPs are built by a chunker. Machines, being more general than FSTs, would also be capable of performing two-level morphological analysis and FST-based chunking, which means that in principle all stages of linguistic analysis could be recast in machine terms. In practice, we didn't see the need to reimplement everything just to make this point, and use preexisting morphological analysis and NP chunking tools. We note that the case grammatical analysis that we rely on means that English NPs also get case marked, the only difference is that we obtain the cases from prepositions and word order (position relative to the verb) rather than from overt morphological marking.

The lexical entries are highly abstract, and for the most part, universal. Thus the verb *go* will have a source and a goal, and the Hungarian lexicon stores that the former can be expressed three different kinds of lative cases (delative *ról/ről*, elative *ból/ből*, and ablative *tól/től*), while the latter can be expressed by illative *ba/be*, sublative *ra/re*, or terminative *ig*. The English lexicon stores the information that source is expressed by the preposition *from* and destination by *to*. Otherwise the two entries are the same, $\underline{\text{before AT SRC, after AT GOAL}}$. Railway-specific entries, such as the names of Intercity trains like *Hírös* and *Feszty* are kept in separate sublexica. Machines are used to encode both lexical entries and constructions in the sense of Berkeley Construction Grammar (CxG, see Goldberg 1995).

The system keeps grammatical and encyclopedic knowledge strictly separated: standard lexical entries store only analytic knowledge about words in the form of dictionary definitions that are written in a small formal language that gets compiled into machine structures at build time. The intricacies of the discount fare structure of the Hungarian Railroad Co. are handled through the AVMs that mediate the interaction between the core system (we hesitate to call it a parser as no parse is generated) and the plugins. The sublexica storing the technical vocabulary are task-specific, but words like *ticket* $\underline{\langle\text{paper}\rangle, \langle\text{card}\rangle, \text{before[pay/812], FOR right/3122}}$ are generic. Space doesn't permit full description of the formal language here, but the commas signify conjunctions (all terms of the definition are conjunctive), angled brackets signify default status (thus, tickets are paper by default) and disambiguation is by numerical indices, so that pay/812 'to pay' is opposed to pay/237 'salary'.

3. SPREADING ACTIVATION

After morphological and phrase-level preprocessing, the input is processed by spreading activation from nodes (machines) that are activated by the input or by the situation until all required slots of at least one AVM are satisfied. The ticket counter situation itself activates the *ticket* and *schedule* nodes and the actual location of the counter, e.g. Budapest Western railroad station. Such situational pre-activation is akin to setting defaults (clearly most users will want tickets from the location of the counter, but they can override this by requesting some other origin), but there is a fundamental difference: there can be no competing defaults, but plugins fulfilling different types of requests (e.g. about schedules, fares, discounts, ordinary tickets or place tickets) are competing with each other.

The architecture itself makes no assumptions about the order or priority of the plugins. For example, in ATIS we may route requests to a pricing plugin via a FareAVM (which has FLIGHT, CLASS attributes) or to a booking plugin with the same attributes, yet the two hardly ever compete, as one will be triggered by the keywords *price* and *cost*, the other by *book* and *reservation*. The system has no dedicated message routing component: wherever the activation is spread is where the action will be. Similarly, there is no separate architecture for context awareness: location detection (by GPS or IP location) may be wrapped into an agent, and this agent may bind the value of *here* to this information. Similar bindings (e.g. setting FARE_CLASS to 1st or 2nd based on whether the user is expensively dressed) are quite conceivable. The overall architecture has the hooks in place for more context-aware extensions, but so far we have not experimented with these.

While the idea of computing the semantics by spreading activation is far from new (for a summary of the early work see Findler 1979), the machine-based system avoids many of the known pitfalls such as the proliferation of link types. In essence, there is only one link, with uniform semantics: A is linked to B iff B appears in the lexical definition of A. When depicting machines as graphs it is convenient to distinguish '1' links (agent, subject, nominative) from '2' links (patient, object, accusative), but there are no '3' (indirect object) or higher links, as ditransitive and higher arity verbs are built by combining intransitive and transitive machines (see Kornai 2012).

The activation algorithm has two phases. In the *construction* phase we take the morphologically analyzed and chunked sentence as input and assemble the machine structures that correspond to its constituents. This is achieved by creating a machine for each word and running them on the chunk the word is in. The type of the machine's FSA correlates with the lexical category of the word. The machines of projective categories attempt to build the corresponding phrase, e.g. from 'the next train' a Noun machine creates train[next]. All words and phrases, whether in the lexicon or the input, are represented by machines. Interaction above the chunk level, such as filling the case frame of verbs, or the slots of CxG constructions, is done in the next, *spreading* phase.

To control spreading we maintain two graphs: a *static* graph whose nodes are the machines corresponding to words and whose edges are the definitional links, and an *active* graph that keeps track of the active machines pertaining to the currently analyzed sentence. For every utterance, full or fragmentary alike, the active graph is initialized from the machine structures created in the construction phase. In each iteration, the active graph is extended and transformed in three stages: *expansion*, *activation*, and *linking*.

Expansion and activation are driven by the lexical definitions of words. In the expansion step we take every, as yet unexpanded, active word, and add to the graph the machine structures compiled from their definitions, each structure connected to the word it defines. This is also how AVMs are activated: each AVM is associated with a word (e.g. TicketAVM with the word *ticket*, GroundTransportAVM directly by the phrase *ground transport(ation)* or indirectly by *limo*), and becomes available only when the corresponding word is expanded. Activation works in the opposite direction: words whose definition structure is a subgraph of the graph of active machines are also added to the graph. Lastly, linking is responsible for filling the empty valency slots of verbs and AVMs. Linking is driven by a handful of explicitly designated linkers (Ostler 1979), corresponding roughly to deep cases.

If not guided carefully, the above algorithm could eventually activate the whole lexicon. To avoid this problem, we apply a heuristic that, based on the structure of the static graph, prefers expanding towards other, already active nodes. The algorithm is run until one of the stopping conditions is met: either we succeed in filling all required slots of an AVM, or exceed a predefined number of iterations; in which case the system notifies the user that it could not complete the request.

4. EVALUATION

The system was implemented in about 8,000 lines of python code and publicly demonstrated in September 2012 – response time is in the subsecond range. Since the MÁV corpus was used during development, it makes little sense to use it for evaluation, but we note that over 60% of the requests were fragmentary both in the corpus and in trials. This is in sharp contrast to ATIS, where most utterances are fully grammatical. In MÁV, only a small fraction (less than 10%) of the requests were informational (can I pay with a credit card, can I take a bike on the train, when is the next train to ...), the rest were ticket, place ticket, and season ticket requests. For now, the system answers questions related to the train schedule, but not those related to MÁV rules and regulations.

ATIS provides an excellent dataset to evaluate a similar English system, with high quality morphological analysis and chunking available in the Penn Treebank . After the collection phase, about 30% of the user input was classified by a committee as 'context-dependent, ambiguous, ill-formed, unanswerable, or noncooperative' . While we have results similar to those reported in the literature, 10-15% error on the remaining 70% , a fair comparison with the more integrated systems is hard, inasmuch as we don't have a speech from end and user queries were typed in manually. That said, the following two points are worth noting.

First, the variety of questions is far broader: in ATIS we needed a total of 9 AVMs compared to 3 in MÁV. In principle, people could ask questions concerning the technical specifications of trains analogous to *What is the wingspan on a 767?* but in practice they don't. The same point could be made in regards to ground transportation, facilities at terminals, meals available, the meaning of abbreviations, etc. Second, in ATIS long (40 minute) conversations with the system were encouraged by the data collection protocol, and there is a definite need for maintaining the results of the inquiry phase for reuse in the booking phase, something that our system currently lacks.

Another, rather disturbing, form of evaluation comes from the lay public. The system described here works well enough for people to oppose its deployment as 'this will only make a few billionaire programmers richer while taking away the job of Aunt Mary' (blog comment on Szedlak:2012).

5. CONCLUSION

It is hard to deny that the system described here goes somewhat against the grain of contemporary computational linguistics. There are hard rules operating on discrete knowledge states, there is no statistical component, and most regrettably, there is no mature technology for the automatic acquisition of the machines/lexical entries that do the work. Yet at the same time the system escapes most of the problems that motivated the shift from discrete symbol manipulation to continuous optimization.

First, and most important, it has legs: the lexical entries used for the MÁV task are in no way specific to this domain and have been reused for ATIS without any significant modification. To be sure, there are many differences between the railroad and the air travel databases, and many differences between these modes of travel – there are no 'open jaw' railroad trips and no 'dining cars' in the air. But the basic conceptual structure, as captured by the Ticket AVM, and the basic syntax of getting from A to B on day C, are shared across these domains. The system presented here is a hardline extension of Karttunen's (1989) Radical Lexicalism: the claim is that once you have taken care of the words, both the syntax and the semantics of a language are completely determined.

Second, it is not particularly brittle: adding new lexical entries to the network does not have deep ramifications for what may be taking place in some other corner. Changes are reasonably localized and debuggable. This is actually an advantage compared to systems with continuous weights, where bad effects (bugs) are impossible to attribute to specific causes.

Third, we are in no way restricted to very experienced programmers who take years to learn the ins and outs of the system. It takes only a few hours of training to teach the monosemic definitional style to undergraduates, comparing quite favorably to the effort it takes to explain e.g. the MUC named entity tagging guidelines . At the same time, the design avoids the known pitfalls of trying to extract everything by surface pattern matchers, an approach that falls apart when faced with the highly fragmentary, free word order user input seen in the MÁV corpus.

This is not to say that we advocate a return to old-fashioned AI methods. To the contrary, we believe that the future is automatic acquisition of lexical entries, and that our current lexicon is but a manually annotated gold standard that future machine learners one day may be evaluated on.

Acknowledgements

The first f our a uthors w rote t he c ode. K ornai a dvised and wrote the paper. Work supported by OTKA grant #82333.

References

Appelt, D. E., Hobbs, J. R., Bear, J., Israel, D. & Tyson, M. (1993), FASTUS: A finite-state processor for information extraction from real-world text, *in* 'Proceedings of IJCAI-93'.

Black, A. W., Burger, S., Conkie, A., Hastie, H., Keizer, S., Lemon, O., Merigaud, N., Parent, G., Schubiner, G., Thomson, B., Williams, J. D., Yu, K., Young, S. & Eskenazi, M. (2011), Spoken dialog challenge 2010: Comparison of live and control test results, *in* 'Proceedings of the SIGDIAL

2011 Conference', Association for Computational Linguistics, Portland, Oregon, pp. 2–7.

Chichor, N. & Marsh, E. (1998), MUC-7 information extraction task definition, *in* 'Proc. Seventh Message Understanding Conference (MUC-7)', ACL, pp. M98–1027.

Eilenberg, S. (1974), *Automata, Languages, and Machines*, Vol. A, Academic Press.

Findler, N. V., ed. (1979), *Associative Networks: Representation and Use of Knowledge by Computers*, Academic Press.

Goldberg, A. E. (1995), *Constructions: A Construction Grammar Approach to Argument Structure*, University of Chicago Press.

Grefenstette, G. (1999), Light parsing as finite state filtering, *in* A. Kornai, ed., 'Extended Finite State Models of Language', Cambridge University Press, pp. 86–94.

Hemphill, C. T., Godfrey, J. J. & Doddington, G. R. (1990), The ATIS spoken language systems pilot corpus, *in* 'Proceedings of the DARPA speech and natural language workshop', pp. 96–101.

Karttunen, L. (1989), Radical lexicalism, *in* M. Baltin & A. Kroch, eds, 'Alternative Conceptions of Phrase Structure', University of Chicago Press.

Kornai, A. (2010), The algebra of lexical semantics, *in* C. Ebert, G. Jäger & J. Michaelis, eds, 'Proceedings of the 11th Mathematics of Language Workshop', LNAI 6149, Springer, pp. 174–199.

Kornai, A. (2012), Eliminating ditransitives, *in* P. de Groote & M.-J. Nederhof, eds, 'Revised and Selected Papers from the 15th and 16th Formal Grammar Conferences', LNCS 7395, Springer, pp. 243–261.

Marcus, M., Santorini, B. & Marcinkiewicz, M. A. (1993), 'Building a large annotated corpus of English: The Penn treebank', *Computational Linguistics* **19**, 313–330.

Ostler, N. (1979), *Case-Linking: a Theory of Case and Verb Diathesis Applied to Classical Sanskrit*, PhD thesis, MIT.

Recski, G. & Varga, D. (2009), 'A Hungarian NP Chunker', *The Odd Yearbook* .

Szedlák, A. (2012), 'Felsögödig kérek egy ilyen nyugdijas', *Origo Techbázis* pp. http://www.origo.hu/techbazis/20120928–felsogodig–kerek–egy–ilyen–nyugdijas–robot–mavpenztarost–epit–a–sztaki.html.

Trón, V., Gyepesi, G., Halácsy, P., Kornai, A., Németh, L. & Varga, D. (2005), Hunmorph: open source word analysis, *in* M. Jansche, ed., 'Proc. ACL 2005 Software Workshop', ACL, Ann Arbor, pp. 77–85.