Bernadette Sharp, Wiesław Lubaszewski
and Rodolfo Delmonte (eds)

Natural Language Processing and Cognitive Science

Proceedings 2015

CAFO
SCAR
INA_

Editors

Bernadette Sharp, Staffordshire University, U.K.
Wiesław Lubaszewski, Jagiellonian University, Poland
Rodolfo Delmonte, Ca' Foscari University, Italy

Natural Language Processing and Cognitive Science
Bernadette Sharp, Wiesław Lubaszewski and Rodolfo Delmonte (eds)

ISBN 978-88-7543-394-9

Libreria Editrice Cafoscarina
Dorsoduro 3259, 30123 Venezia
www.cafoscarina.it

Prima edizione luglio 2015

Contents

Preface

The 12th workshop on Natural Language Processing and Cognitive Science (NLPCS 2015) is a forum for researchers and practitioners in Natural Language Processing (NLP) interested in taking a Cognitive Science perspective and learning from recent advances in Cognitive Neuroscience, Cognitive Linguistics and Neurolinguistics. This workshop was held on 22-23 September, 2015 in Krakow, hosted by the Department of Computational Linguistics, at Jagiellonian University.

The NLPCS workshops have attracted computer scientist, computational linguist and cognitive linguist researchers from all over the world. In addition to the proceedings the workshops contributed to the following two international journal issues and three books:

Special issue of the International Journal of Speech Technology, vol. 11, issue 3/4, December 2008

Special issue of the International Journal of Speech Technology, vol. 12, 2/3, September 2009

Gala, N., Rapp, R. & Bel-Enguix, Gemma (Eds.) Language Production, Cognition, and the Lexicon, Springer, 2014

Neustein, A. & Markowitz, J.A. (Eds.) Where Humans Meet Machines: Innovative Solutions to Knotty Natural Language Problems, Springer Verlag, Heidelberg/New York, 2013

Sharp, B., Zock, M., Carl & M. Jakobsen, A. L. (Eds.) Human Machine Interaction in Translation, Copenhagen Studies in Language, Vol. 41, 2011

The papers and posters presented at the workshops covered an impressive range of approaches ranging from linguistics, cognitive and computer science study to language processing. They covered a wide variety of languages including Arabic, Polish, Czech, Bulgarian and Mizo languages.

We would like to thank the authors for providing the content of the programme. In particular thanks to our invited speaker, Philippe Blache, who has contributed the paper entitled "Hybrid Parsing for Human Language Processing". We are grateful to the programme committee who worked very hard in reviewing papers and providing helpful feedback to the authors. We would like also to thank Jagiellonian University for hosting the workshop and for their help with housing and catering. Special thanks to Maciej Godny for his help with the administrative support of NLPCS website. Thanks in particular to Stefano Chinellato for his help in publishing the proceedings.

We hope that you will find the proceedings interesting and thought-provoking and that the workshop will provide you with a valuable opportunity to share ideas with other researchers and practitioners from institutions around the world.

September 2015
Co-chairs of the workshop:
Bernadette Sharp, Staffordshire University, U.K.
Wiesław Lubaszewski, Jagiellonian University, Poland
Rodolfo Delmonte, Ca' Foscari University, Italy

Hybrid Parsing for Human Language Processing[1]

Philippe Blache

Aix-Marseille Université & CNRS, Laboratoire Parole & Langage, Aix-en-Provence, France
blache@lpl-aix.fr

Abstract. This paper presents an architecture for human language processing, explaining both facilitating and complexification effects. It relies on the hypothesis that the default strategy is shallow parsing and relies on chunks (or constructions), considered as basic units. The paper proposes a representation of these units in terms of properties. It presents then an algorithm schema, in which a hybrid technique, integrating shallow and deep parsing, is described.

1 Introduction

A very common observation when studying human language processing is that it is an extremely fast process, in spite of the apparent complexity of the task. However, this characteristics remains largely unexplained. As an illustration, many studies in psycholinguistics have explored parameters that render parsing difficult (see for example Gibson, 2000; Grodner et al., 2003). However, very few works tries to identify what facilitates processing (Blache, 2011).

We propose in this position paper a language processing architecture based on the idea that the default processing by human is very superficial. We only do in most of the cases a simple shallow parsing that is usually enough to understand what we read or hear. Our hypothesis more precisely is based on the idea that the processing relies on the identification of intermediate groups that are the basic operational units, instead of words. Processing an input consists then basically in identifying these units. Moreover, the hypothesis also predicts that the presence of such groups facilitates processing. In this paper, we will first give a description of the basic operational units in terms of *constructions*. We propose then a formal approach for their representation by means of *properties*, starting from which constructions can be recognized. The shallow parsing technique is then described, before presenting the general processing architecture.

[1] Research supported by grants ANR-11-LABX-0036 (BLRI) and ANR-11-IDEX-0001-02 (A*MIDEX)

2 The operational level: chunks, constructions

Classical architecture of human language processing relies on the hypothesis of on an incremental word-by-word integration. However, several observations militate in favour of the existence of intermediate operational units. We present in this section a brief overview of the notions of chunks and constructions, that can play such a role.

2.1 Chunks

Chunks are broadly used in natural language processing (Abney, 1991; Bird, 2009) as well as psycholinguistics (Anderson, 2003). Chunks are especially relevant in designing shallow parsing mechanisms, used for different language processing tasks. They are defined as group of words or categories that are identified by means of local and low-level properties. It is a non-recursive structure, gathering the words that are tightly connected and adjacent. Classically, chunks are recognized starting from their boundaries: left corner (thanks to the intrinsic properties of certain categories such as the determiner or the preposition) or right bound (thanks to the transition between two adjacent categories). This boundary recognition is done very efficiently by means of probabilistic techniques: n-grams directly model such transition properties. Chunks can also be identified on linguistic basis as set of words with a strong syntactic relation (for example between a *specifier* and a *head*). Such symbolic definition of chunks has been used for example in the context of parsing evaluation (Paroubek et al., 2008), in which chunks has been defined in terms of syntactic units, gathering the main adjacent constituents of the different phrase types.

In a cognitive perspective, different studies have shown the importance of chunks in human language processing. For example, (Krishnamurthy, 2003) suggests that words are not the *operational unit* in language processing when learning a language, the real unit being chunks, defined as groups of words that form meaningful units. In the same vein, another study has shown using oculometry that chunks are also more likely to be basic units when reading (Rauzy et al., 2012). One observation that can be done is that the presence of chunk is a facilitator of language processing: chunks are read faster than unlinked set of words (Ellis, 2003). In this perspective, several studies in neuroscience have identified a functioning based on chunks, treated as *lexical units* (Capelle et al., 2010).

2.2 Constructions

The notion of *construction* in grammar (Fillmore, 1995) relies on a specific form-function relation coming from the convergence of different properties (lexical, semantic, syntactic, etc.). Constructions are patterns in which the meaning emerges from the interaction between the different components, not compositionally (Goldberg, 2009). The following examples illustrate different types of constructions:

1. Covariational Conditional
   The Xer the Yer (e.g. "The more you watch the less you know")

2. Ditransitive
   Subj V Obj1 Obj2 (e.g. "She gave him a kiss")
3. Idioms
   e.g. "kick the bucket"

In construction-based approaches, language (or more precisely speakers' knowledge of language) is based on collections of such form-function pairings. An important question in the perspective of language processing is to understand how constructions are recognized. When taking the case of idioms in which meaning is completely non-compositional, psycholinguists propose two different solutions. One is the "*Lexical Representation Hypothesis*" (Bobrow et al., 1973). In this case, idioms are stored with normal words in memory. They are processed both literally and figuratively simultaneously, but the figurative meaning is accessed first. However, this explanation does not account for idiom flexibility: many of them can be transformed to some extent and still be recognized and understood as idioms. It does not also explain the fact that idioms are processed more rapidly than literal expressions.

In other approaches, idiom processing, instead of being lexical, relies on "normal" language processing. This is the case of the "*Configurational Hypothesis*" (Cacciari et al., 1988) in which a sufficient portion of an idiomatic expression must be processed literally before the idiom can be identified. After reaching this recognition point, the rest of the idiom is not processed literally anymore. It has been shown in particular that the brain activity differs before and after the idiom recognition point: one event-related potential, called N400, shows a lower negativity (Vespignani et al., 2009).

At a more general level, several studies have started to investigate neurological evidence of construction role during language processing, stipulating the existence of constructional templates in the brain (Pulvermuller et al., 2013). As it is the case with chunks (and for the same reasons), the identification of a construction seems to play a facilitator role in language processing, which is observed both in time-reading and brain activity.

3 Representing linguistic properties

Classical incremental approaches do not integrate easily constructions into a general processing architecture. The first problem is that we have to involve into a unique mechanism different types of objects (words, categories, constructions). Moreover, the recognition of a construction relies on the accumulation of different properties, and sources of information.

We propose in our approach an important shift: instead of building compositionally a semantic representation starting from the different linguistic domains (especially syntax), we propose to start by gathering all possible information, and then trying to see how they can lead to an interpretation. Understanding a sentence (or a message) does not consist in building a structure (for example a formula), but in identifying the level of information available for the interpretation. We propose for this to represent separately all the different types of information, also called *properties* (Blache, 2000), whatever their level (relations between features, categories, chunks, etc.) or their domain (morphology, syntax, semantics, etc.). These properties connect the different

words of a sentence when processing an input. In this approach, instead of building a structure, the processing mechanism consists in describing the input by identifying the different properties.

Starting form syntax and semantics, here is a list of possible properties that can be directly represented as relations between words:

- *Linearity*: linear order that exists between two words
- *Co-occurrence*: mandatory co-occurrence between two words
- *Exclusion*; impossible co-occurrence between two words
- *Uniqueness*: impossible repetition of a same category
- *Dependency*: syntactic-semantic dependency between two words. Different types of dependencies are encoded: complement, subject, modification, specification, etc.

A grammar is a set of all the possible relations between categories, describing the different constructions. In terms of operational semantics, the interpretation is straightforward. Given a sentence *S*, evaluating a property of the grammar consists in verifying whether the relations between two categories corresponding to words of *S* are true. For example, *linearity* consists in checking the linear order between the categories of the corresponding words in the sentence to be parsed. As another example, *uniqueness* verifies that a same category, within a set of categories corresponding to a subset of words in a sentence, is not repeated.

The following figure shows an example in which the different properties are represented by labelled edges[2] between words:

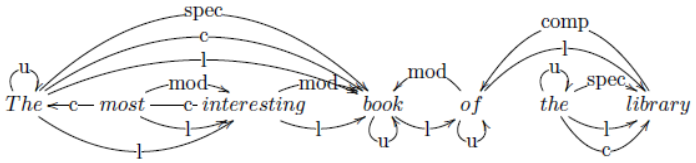


Fig. 1. Graph of properties describing a sentence

As can be seen in this graph, some subsets of words are more connected between each other. The idea is that this density represents a higher level of information, corresponding to constructions[3]. A construction being a subset of words of the sentence, the set of constructions forms a partition of the sentence.

A property is a relation of a certain type, that can be unary or binary. Moreover, such relation can be more or less imperative, corresponding to the distinction between hard and soft constraints (Keller, 2010). This information is implemented in terms of weights (in the following examples, we use *H+, H* and *S* values, distinguishing between hard and soft properties). At this stage, a property is a tuple of the form:

```
<id, relation-type, source_node, target_node, weight>
```

The following example illustrates the case where some relations are of particularly high weight. This is typically the case of idiomatic construction: after the recognition

[2] Edges are labelled with the type of the property (co-occurrence, linearity, dependency, etc.)
[3] In this representation, all connected subgraphs correspond to syntactic constructions.

point, the co-occurrence as well as linearity relations become imperative constraints. They are represented in the graph with a double arrow:

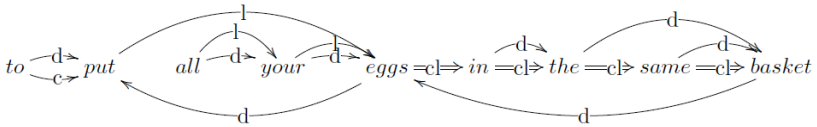


Fig. 2. Constraint graph for an idiomatic construction

All properties are represented independently from each other. However, as described in the previous section, constructions correspond to set of properties that interact together. It is then necessary to represent such links between properties describing the same construction. At the difference with constituency-based approaches in which a construction is described in terms of sets, our approach consists in specifying directly the links between the properties. We propose to add to the representation of the properties this interconnection information by adding a new argument encoding the properties linked to the current as follows:

```
<id, type, source, node, weight, linked_props>
```

The `linked_props` argument is a set of indexes, pointing towards other relations describing the same construction. For example, the dependency relation between a *preposition* and a *noun* depends on the linearity: if *Prep*<*N*, then *Prep* is the head and *N* depends on it. Reciprocally, when *N*<*Prep*, the *Prep* depends on N. These relations between properties are represented as follows

`<1, lin, Prep, N, H, {}>`	`<2, comp, N, Prep, L, {1}>`
`<3, lin, N, Prep, H, {}>`	`<3, mod, Prep, N, L, {3}>`

The example of the *ditransitive construction* can be implemented in the same manner, specifying different dependency types according to the form (the first noun is the indirect object, the second the direct):

`<1, lin, V[dit], N1, H, {}>`	`<4, iobj, N1, V, H, {1,2,3}>`
`<2, lin, V[dit], N2, H, {}>`	`<5, obj, N2, V, H, {1,2,3}>`
`<3, lin, N1, N2, H, {}>`	

In this example, the properties implementing the dependency relations are linked to the three first properties with the corresponding indexes.

4 Two types of parsing with properties

4.1 Shallow vs. Deep Parsing

A property can be evaluated by a function returning its truth value. Two types of properties can be distinguished according to the way they are evaluated (Blache & Dahl, 2004):

- **Success-monotonic properties**: when a relation between two categories holds, it remains true when parsing the rest of the sentence. For example, the linearity between *most* and *interesting* in Fig. 1 holds a soon as it can be evaluated, and remains true until the end. In a more formal manner, the linearity relation $a<b$ is true in the sequence of words $s=[\gamma, a, b, \eta]$, whatever the composition of γ and η. Two types of properties are success-monotonic: linearity and co-occurrence.
- **Success-non monotonic properties**: A property can be true locally and false at a larger span: the evaluation of a property depends on the set of categories taken into account. For example an exclusion relation between he words *a* and *d* is true within the set of words *s1={a, b, c}*, but false when adding a new category *d* to this sequence *s2={a, b, c, d}*. In this case, it is then necessary to choose a partition into which evaluating the constraint.

We propose to distinguish two kinds of parsing based on a distinction between shallow and deep parsing. Moreover, we want this distinction to be cognitively grounded in terms of memory load: shallow parsing does not require much resource, contrarily to deep. The above distinction of monotonicity makes it possible to implement the parsing strategies:

- **Shallow parsing**: evaluation of *success-monotonic properties*. In this case, properties can be evaluated independently from the context. There is no ambiguity, the evaluation is independent from the subset of categories of the sentence.
- **Deep parsing**: evaluation of *all properties*, monotonic or not. In this case, it is necessary to take into consideration all the possible partitions of the sentence. (i.e. the possible subsets of words of the sentence). In a classical view, this comes to explore all the possible solutions.

The general parsing process consists in evaluating properties when scanning a new word in the sentence. This evaluation consists in identifying in the grammar all the properties having the word or its category as a target.

In the case of *shallow parsing* (i.e. evaluation of *linearity* and *co-occurrence*), all properties can be directly evaluated (no need of the context). It is interesting to note that these two relations are the one that are crucially encoded by *n-grams* used in a probabilistic approach. In this case, a transition probability between two categories is higher when strong linearity and co-occurrence relations link them. This characteristic will be of certain interest for the specification of the parsing architecture.

In the case of *deep parsing*, the mechanism consists in building the different partitions of the set of words from the beginning until the word to be integrated, and then evaluating the properties having the current word as target, taking into account the subset in which it appears.

4.2 Inferring Properties

Our representation of constructions underlines the interaction existing between the properties. More precisely, some properties of the constructions are specific in the

sense that they depend on the realization of other properties. For example, the grammatical role of nominal objects in the ditransitive construction depends on the linearity of the constituents. We can observe the same kind of relation in most of the construction. In particular, it is very often the case that dependency properties depend on the realization of other properties. This also means that some properties can be predicted from a set of properties already evaluated.

For example, as seen above, the dependency relation between a prepositional and a nominal construction depends on the linearity between them. We can then *infer* their dependency relation from the evaluation of linearity. The same type of inference can be applied in many other cases. For example it is also possible to infer dependency relations from the linearity ones in the ditransitive construction. In such cases, a subset of properties acts as a trigger for targeted ones.

Moreover, inference can also be applied to modify or precise some features or values of the properties within a construction. For example, in the case of an idiom, the weights of the co-occurrence and linearity relations between the words can be inferred after reaching the recognition point. In the same way as for triggering properties, the sequence of words until the RP is the trigger of the new weights of the rest of co-occurrence and linearity relations.

Inferring new properties or weights is then a direct mechanism, which does not require any analysis process (then any extra cognitive load). Moreover, the triggering properties are in most of the cases directly evaluable (i.e. the success-monotonic). This means that such inference can be done even when doing shallow parsing. In a cognitive perspective, this means that such information only represent a light load: inference simply consists in instantiating new information, completing the existing one. We call this mechanism "*complemented shallow parsing*".

5 Hybrid Parsing

Several works in cognitive psychology use the old hypothesis that working memory has a capacity of seven units (Miller, 1956). This hypothesis has been often challenged (in particular, this capacity seems not to be constant), but the idea remains that when processing an input, we can store a limited amount of information. Without taking position on this debate, our architecture relies on this first idea that a limited amount of memory units, called *buffers*, can be used during parsing.

Other works in this same domain have shown that the kind of information to be bufferized can be complex (Anderson, 2004): several atomic elements can be aggregated and form *chunks*. Our approach also takes this idea that when possible, atomic elements (i.e. words) can be aggregated into chunks. The hypothesis is that these chunks (as presented in the first section) can have different forms and constitute a facilitator to processing.

5.1 Identifying chunks: the notion of cohesion

The question with this schema is how to identify a chunk. As seen above, a chunk is a convergence of a highly cohesive set of words. Such cohesion can be identified thanks to the strength of the relations that linked them together. The cohesion can come from any domain and then be of different types, for example:

- *Lexical selection*: in the case of multiword-expressions or frozen idioms, there exists a co-occurrence and linearity properties that link the words together. These properties bear the maximal weight. Moreover, these properties also makes it possible to directly infer a semantic interpretation, that reinforce cohesion of the set.
- *Subcategorization*: several constructions are based on a mandatory subcategorization of the complement by the governor. Typically, certain verbs are necessarily transitive and can never be constructed in an intransitive manner. In this case too, this is represented by linear and co-occurrence properties with a heavy weight.
- *Constructions*: most of the constructions are the result of the convergence between a large number of properties. In this case, each property is not necessarily of a heavy weight. The cohesion comes from the density of the property network.

The basic mechanisms when building a chunk consists then in evaluating the cohesion of the set of words, thanks to density and weights. For doing so, we propose a simple cohesion function, based on two factors: density and weights, defined as follows:

$$cohesion = \frac{\sum prop\_weights}{|words|}$$

In this formula, density is the sum of the property weights divided by the number of words. In this approach, both density (number of properties) and weights (relative importance of a property) are taken into consideration.

For any set of words, it becomes then possible to evaluate directly its cohesion. The decision whether a set of words forms a chunk or not depends then from the choice of a *threshold*, beyond which the structure is considered to be highly cohesive.

5.2 The processing schema

The processing architecture relies on two types of processing that are applied depending on the input. In the general case, we only use *complemented shallow parsing* to identify the possible chunks/constructions. It can be the case that this mechanism leads to a complete processing of the sentence and its interpretation. In some situations, it is not possible to fully integrate all the elements into a unique chunk. In this case, we apply then a *deep parsing* technique, dealing with ambiguity and exploring the possible interpretations.

Concretely, at each step (i.e. at each new item scanned from the input), a *complemented shallow parsing* is applied to the current sequence of words. This sequence is

made of the last chunk under construction (possibly made of a unique word) plus the new input word. If the sequence reaches a certain cohesion threshold (see previous section), then the input word is aggregated to the current chunk. This mechanism can be seen as a *shift/reduce* processing: when possible, a sequence of words is merged into a single unit, applying there a *reduce* operation.

It is important to remind that, even when using shallow parsing, new properties can be inferred, in particular the dependency ones. In this case, we can start to build at this basic level a semantic structure. The semantic aspect is an important parameter coming into play when identifying a chunk. As explained, a chunk is a cohesive set of words. This cohesion can be identified in some cases only thanks to syntactic constraints such as linearity, exclusion, etc. When a chunk also bears semantic information (a dependency structure leading to an interpretation), then it constitutes a construction. In the –extreme- case of idioms, the identification of the chunk as well as its interpretation comes directly (not compositionally) after the recognition point.

The general process consists then in scanning the entire input, trying to reduce as much as possible into chunks. In the cognitive architecture, a chunk occupies a unique *buffer*. When a new word cannot be integrated into the current chunk, it is then stored into a new buffer. This mechanism is applied until reaching the maximal capacity of the working memory (let's say seven buffers). Reaching this limit means that no reduction can be done for the set of words, and no interpretation can be given. In such a situation, a deep parsing process is launched, exploring incrementally all the possible structures leading to an interpretation. This means to explore different possible solution, trying to identify the optimal one.

An algorithm schema can be given, presenting the main lines of this hybrid parsing. In this schema, we have a stack of buffers storing words or chunks. What is stored is more precisely the property graph associated to the words (i.e. the set of words plus their properties). In the following, we note functions in italics with an initial capital letter and data structure in lower case. The function `Scan` returns the current word of the input sentence.

```
Init:
     i=0; j=0; buffer(bj) ← Scan(wi)
repeat
     i++; Scan(wi)
     ci ← bi-1 + wi
     graph[ci] ← Shallow_parse(ci)
     if Cohesion(graph[ci]) > threshold
     then buffer(bj) ← graph[ci]
     else j++; buffer(bj) ← wi
until (j=7 or eos)
if (j>1) then Deep_parse([b1..bj])
```

The schema consists in a loop trying to identify the chunks thanks to shallow parsing. Chunks (ore isolated words when no aggregation is possible) are stored into buffers. When the buffer stack reaches the maximal memory capacity, a deep parsed is

launched. This schema makes it possible to identify different situations, correlated with different processing difficulty levels:

- Simple processing: the entire input can be reduced into a unique chunk (at the end of the process, the buffer stack contains only one buffer with one chunk). Accessing to interpretation is done only by means of shallow parsing.
- Medium difficulty: several chunks can be identified; the final interpretation process relies on deep parsing integrating the different chunks. The overall process makes use of shallow and deep parsing.
- Difficult processing: no chunks can be identified. The only process relies on deep parsing.

6 Conclusion

In this paper, we have explored the idea that the default mechanism in human language processing is shallow parsing. In most of the case, starting from very basic properties, higher-level information can be inferred, until reaching the possibility to a direct access to the meaning of entire subparts of the sentence, without any need of complex compositional mechanism. This architecture relies on the existence of intermediate operational units formed by constructions. These elements are form-meaning pairings, identified by a convergence of different linguistic properties. Very often, the constructions can be recognized starting from basic properties. As soon as a construction is recognized, the corresponding meaning can be accessed directly. The presence of constructions is then an important facilitator effect.

We have proposed a parsing strategy implementing a hybrid parsing: shallow parsing as default, and deep parsing when no construction can be built. This strategy is in line with the cognitive architectures, describing the working memory as a set of buffers. When a construction is recognized, it is stored in a buffer that contains otherwise only words). The number of buffer is limited (many approaches evoke the number of 7 buffers). When the maximum capacity of the memory is reached, then a deep parsing is applied.

This hybrid processing architecture offer a framework explaining both facilitating and complexification effects during language processing. Moreover, the property-based representation provides the basis of a new shallow processing technique, explaining how new information (in particular meaning) can be accessed directly.

References

Abney, S. (1991) Parsing by chunks. In Principle-Based Parsing. Kluwer Academic Publishers

Anderson, J. R., Bothell, D., Byrne, M. D., Douglass, S., Lebiere, C. Et Qin, Y. (2004) An

integrated theory of the mind. Psychological Review, 111(4)

Bird S., Klein E., Loper E. (2009) Natural Language Processing with Python - Analyzing Text with the Natural Language Toolkit, O'Reilly Media

Blache P. (2000) Constraints, Linguistic Theories and Natural Language Processing, in Natural Language Processing, D. Christodoulakis (ed), Lecture Notes in Artificial Intelligence 1835, Springer-Verlag

Blache P. (2011) "A computational model for linguistic complexity", in proceedings of the first International Conference on Linguistics, Biology and Computer Science

Blache P. & Dahl V. (2004), Directly executable constraint-based grammars, in proceedings of JFPLC-04

Bobrow, S. A., Bell, S. M. (1973) On catching on to idiomatic expressions, Memory and Cognition, 1:3

Cacciari C. & Tabossi P. (1988) The comprehension of idioms, Journal of Memory and Language, 27:6

Cappelle B., Shtyrov Y, Pulvermüller F. (2010) Heating up or cooling up the brain? MEG evidence that phrasal verbs are lexical units, in Brain & Language 115

Ellis, N. C. (2003) Constructions, chunking and connectionism: The emergence of second language structure. In C. J. Doughty & M. H. Long (Eds.), The handbook of second language acquisition, Blackwell Publishing.

Frank A., Becker M., Crysmann B., Kiefer B. and Schäfer U. (2003) Integrated Shallow and Deep Parsing: TopP meets HPSG, in Proceedings of the 41st Annual Meeting of the ACL

Gibson, E. (2000) The dependency locality theory : A distance-based theory of linguistic complexity. In Image, language, brain. A. Marantz, Y. Miyashita, W. O'Neil (eds), MIT Press

Grodner, D. and Gibson, E. (2005). Consequences of the serial nature of linguistic input for sentential complexity. Cognitive Science, 29

Hammerton J., Osborne M., Armstrong S. (2002) Introduction to Special Issue on Machine Learning Approaches to Shallow Parsing, in Journal of Machine Learning Research 2

Krishnamurthy, R. (2003) Language as chunks, not words, in M. Swanson, & K. Hill (Eds.), in proceedings of JALT2002 : conference proceedings: waves of the future

Miller, G. A. (1956) The magical number seven, plus or minus two: Some limits on our capacity for processing information, in Psychological Review 63 (2)

Paroubek P., Robba I., Vilnat A. and Ayache C. (2008) EASY, Evaluation of Parsers of French: what are the results? , in proceedings of LREC 2008

Pulvermüller F. , Cappelle B.and Shtyrov Y, (2013) Brain Basis of Meaning, Words, Constructions, and Grammar, The Oxford Handbook of Construction Grammar, T. Hoffmann and G. Trousdale (eds), Oxford University Press

Vespignani F., Canal P., Molinaro N., Fonda S., and Cacciari C. (2010) Predictive Mechanisms in Idiom Comprehension, in Journal of Cognitive Neuroscience 22:8

An Evaluation of AMIRA for Named Entity Recognition in Arabic Medical Texts

Saad Alanazi[1, 2], Bernadette Sharp[2] and Clare Stanier[2]

[1] College of Computer Science and Information, Aljouf University, Skaka, Saudi Arabia
saad.alanazi@research.staffs.ac.uk

[2] Faculty of Computing, Engineering and Technology, Staffordshire University, Beaconside, Stafford ST18 0AD, UK
{B.Sharp, C.Stanier} @staffs.ac.uk

Abstract. A study is carried out to evaluate the AMIRA tool which has been used widely to pre-process Arabic texts for natural language processing tasks. AMIRA is used in our study to tokenise and POS tag our Modern Standard Arabic medical texts. AMIRA includes a tokeniser, POS tagger, and a base phrase chunker. The AMIRA tokeniser has achieved 91.22%, 87.15% and 89.13% for precision, recall and F-measure, respectively, while AMIRA POS tagger achieved 84.09% accuracy. The most common errors in the tokeniser outputs were in the words where the first letter after the ال (Al) determiner is ل (L). With respect to the POS tagging, AMIRA underperformed in the following categories: broken plurals, adverbs, adjectives and genitive nouns.

1 Introduction

The term "Named Entity", which was coined for the Sixth Message Understanding Conference (Grishman & Sundheim 1996) was initially applied to information extraction tasks aimed at extracting names of person, organisation and locations as well as numeric and percent (e.g. time, date, money) expressions from structured and unstructured documents. This task was not only recognised as essential step of information extraction but became a focus of study for many researchers.

This paper focuses on text tokenisation and part-of-speech tagging (POS), two crucial steps in many natural language processing applications and, in particular, in named entity recognition. The first task is tokenisation which aims to convert text into tokens, where tokens are one or more characters that express an independent linguistic meaning, and roughly correspond to words. The tokenisation task is crucial because errors made in this phase can propagate into later phases and lead to serious problems. It may seem less challenging in the context of some languages, such as English, where a single space or punctuation is used to split sentences into words (tokens). However, it is very challenging in some languages, like Chinese, Japanese, and Thai, which do not use spaces to split sentences into words (Peng et al., 2004). It is a challenging and non-trivial task in the Arabic language as word tokens cannot be delimited solely by a blank space because Arabic words are often ambiguous in their morphological structure. The

aim of the second task is POS tagging which assigns an appropriate POS tag to every token in the input data (Voutilainen, 2003). As Arabic has a very rich and complex morphology a word can carry not only inflections but also clitics, such as pronouns, conjunctions, and prepositions. A single stem may correspond to thousands of different word forms (Habash, 2010; Mohamed & Kübler, 2010).

The aim of our research is to extract information about symptoms, treatment and drugs relevant to cancer from Arabic medical literature. We have used the AMIRA tool developed at Stanford University (Diab, 2009) in our tokenisation and POS tasks. This paper discusses the problems and issues encountered in applying AMIRA. Section 2 explains the challenges related to tokenisation and POS of Arabic texts. Section 3 reviews previous work and section 4 describes the data set, the experimental set up and discusses the results. Section 5 presents our final findings.

2 Challenges of Arabic Language Processing

Arabic has many traits which, make building an effective tokenising and POS tagging tool a very challenging task. Some of these main challenges are described below.

2.1 Agglutination

The Arabic language has an agglutinative nature and this results in different patterns, which can create many lexical variations. It has a very systematic, but complicated morphology. This is seen with words that comprise prefixes, a stem or a root, and sometimes even more than one, as well as suffixes with different combinations. There are also clitics, which in most languages, including English, are treated as separate words; however in the Arabic language, they are agglutinated to words (Farghaly and Shaalan, 2009). For instance, a phrase in English, such as "and they will write it" can be split into five tokens, while in Arabic this is expressed in one word وسيكتبونها (wsyktbonha). As this example demonstrates, the conjunction “and” and the future marker “will” are represented as prefixes by the letter و and س, respectively, while the pronouns “they” and “it” are represented by the suffixes ون and ها, respectively. Because of the complex morphological structure of the Arabic language, the tokenisation process is a difficult and challenging task.

2.2 Short Vowel Absence

Diacritics can be found in Arabic text, which is a representation of most vowels that affect phonetic representation. This lends an alternative meaning to the same word. Consequently, disambiguation in the Arabic language is a difficult task because it is may be written without diacritics (Alkharashi, 2009). For instance, the word كتب without using diacritics could mean the noun “books” or the verb “to write”; therefore, determining the appropriate POS tag is difficult in the absence of diacritics.

2.3 Rich Morphology

Arabic has a very rich morphology. As a result, a vast number of words can be derived from only one root. For instance, the following words have been derived from the root ك ت ب (k t b): كتب (wrote), كتاب (book), كاتب (writer), كتبة (writers – broken plurals), كُتّاب (writers – broken plurals), مكتب (office),مكتبات (offices), مكتبة (bookstore), مكتوب (written), كُتيب (booklet), كاتبون (writers- masculine), كاتبات (writers- feminine), كتيبة (Battalion), and so on. Consequently, the tag set can potentially be huge and can reach over 330,000 tags for untokenised words (Habash, 2010), an additional challenge for Arabic POS tagging.

3 Previous Research

The tokenisation process is often discussed as a part of several existing morphological analysers, such as the Buckwalter Arabic morphological analyser (BAMA), AMIRA (Diab, 2009), MADA+TOKAN, Khoja stemmer and the tri-literal root extraction algorithm (Al-Shalabi et al, 2003). BAMA uses pre-stored dictionaries of words, stem and affixes constructed manually, as well as truth tables to determine their correct combinations (Buckwalter, 2004; (Buckwalter, 2002). BAMA consists of three parts: lexicon, compatibility tables, and an analysis engine. All the prefixes, suffixes, and stems are gathered in a different lexicon. The task of the compatibility table is to determine whether the morphological units (prefix-stem- suffix) are permitted to occur all together or not. The analysis engine produces different morphological analyses such as POS tag, lemma, and morpheme analyses. AMIRA and MADA, both use a support vector machine (SVM) to perform the tokenisation of Arabic words. The AMIRA tool (Diab, 2009) which was developed at Stanford University, includes a tokeniser, POS tagger, and a base phrase chunker. AMIRA uses a fixed size window of +/- five letters; all letters tags within the window are used as features to feed the SVM algorithm. AMIRA provides the user with a choice of three tagging schemes: Bies, ERTS, and ERTS_PER tag sets. In the MADA+TOKAN system MADA which is the morphological analyser makes use of orthogonal features and a list of potential analyses provided by BAMA to select the most appropriate analysis of each word. TOKAN uses morphological generation to recreate the word after splitting off its clitics (Habash et al., 2009). In the Khoja stemmer (Khoja, 1999), the longest prefix and suffix are removed from the word, and then the remainder of the word is matched with the patterns of different nouns and verbs. The stemmer makes use of a list of all diacritic characters, punctuation characters, definite articles, and stop words (Larkey & Connell 2001). Al-Shalabi et al. (2003) have developed a tri-literal roots extraction algorithm that does not depend on any pre-stored information, but assigns mathematical weight to the position of the letters in a word. Higher weights are assigned to the letters at the beginning and at the end of the word and lower weights to root letters.

A comparative analysis of the three stemmers, Khoja stemmer, BAMA, and tri-literal root extraction algorithm, was carried out by Sawalha and Atwell (2008). These three systems were applied to two distinct documents: a newspaper and a chapter

from the Qur'an, each containing about 1000 words. The three stemming algorithms have generated correct analysis for simple roots that do not require detailed analysis. The performance is computed using a majority voting procedure in selecting the most common root among the list of words and their roots. Their analysis showed about 62% average accuracy rate for Qur'an text and about 70% average accuracy for newspaper text.

4 Experimental Study with AMIRA

The accuracy of the stemmers may not be an important issue for information retrieval systems but it is vital for named entity recognition applications. Our approach to extracting specific named entity from cancer documents consists of four main stages: pre-processing, data analysis, feature extraction, and classification stages. The pre-processing stage (in dashed line) covers the data tokenisation and POS tagging approach, which is the focus of this paper. The resulting tokens and their grammatical tags are transferred into a set of features which are then used as inputs for the classification phase. It is proposed to use Bayesian Belief Network to train and classify the extracted features which will then become the recognised entities. Any errors encountered in the early processing of texts have to be rectified to avoid their propagation in subsequent tasks and to produce a reliable training system. Figure 1 displays our named entity recognition system architecture.

In order to perform the text tokenisation task, the AMIRA tool was used as it accepts raw Arabic texts as input and allows the user to choose between different tokenisation schemes.

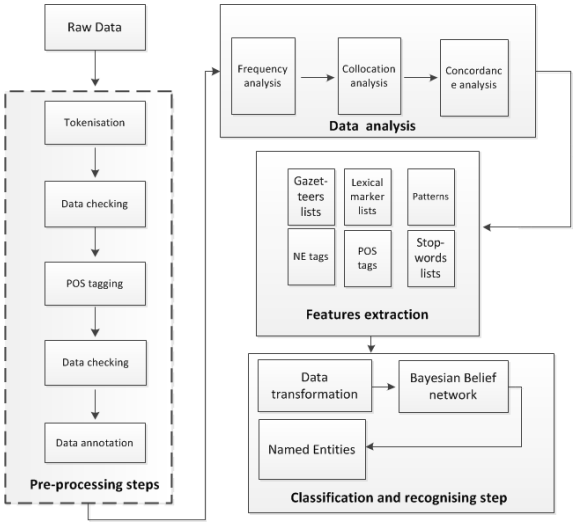


Fig. 1. The NER system architecture

4.1 Data Description

The data for our study is based on Modern Standard Arabic texts extracted from the King Abdullah Bin Abdulaziz Arabic Health Encyclopaedia (KAAHE) website. KAAHE was initiated through the collaboration between the King Saud Bin Abdulaziz University for Health Sciences and the Saudi Association for Health Informatics and further developed by the National Guard Health Affairs the Health on the Net Foundation and the World Health Organisation. KAAHE became the official health encyclopedia in May 2012 (Saudi E-health Organisation, 2012). KAAHE is a reliable health information source, contains abundant information written in an easily understandable language appropriate for users from various community groups (Alsughayr, 2013).

4.2 Tokenisation Task

AMIRA was applied to 26 articles with a total of 5119 tokens. Each article is related to a specific type of cancer. AMIRA allows the user to determine the tokenisation scheme from the different existing schemes. Different prefixes such as conjunctions, future markers and prepositions are selected to be split into parts. The Al determiners and suffixes are not tokenised because this increases the ambiguity and sparsity of the text, as there are more than 127 suffixes in Arabic (Sawalha and Atwell, 2009). Figure 2 displays a sample of the tokenisation result where errors are highlighted in grey.

سرطان الرحم هو السرطان الذي يبدأ في الرحم. يهتم هذا البرنامج بالنوع الأكثر شيوعا من سرطان الرحم، وهو السرطانة البطانية الرحمية. تبدأ السرطانة البطانية الرحمية في بطانة الرحم الداخلية. تنتشر الخلايا السرطانية إلى أجزاء مختلفة من الجسم عن طريق الأوعية الدموية و القنوات الالمفية. و يكون من المستحيل تحديد السبب الدقيق ل الإصابة بالسرطان لدى مريض ب عينه عادة	Uterine cancer is cancer that begins in the uterus. This program will focus on the most common type of uterine cancer, which is endometrial cancer. Endometrial cancer begins in the lining of the uterus. Cancerous cells spread to different parts of the body through blood vessels and lymph channels. It is usually impossible to specify the cause of cancer in an individual patient.

Fig. 2. A sample of the tokenization task result

In the above example, AMIRA missed tokenising the words: بالنوع (*bAlnwE* - type) and بالسرطان (*bAlsrTAn* – by cancer) which starts with the preposition ب (b) and the word وهو (*whw* – and it) which starts with the conjunction و (w). On the other hand, AMIRA tokenised the word اللمفية (*Allmfyp* -lymphatic),which does not need to be tokenised, by adding ا (A) letter after the determiner ال (Al) so the wrong result of tokenisting this word is الالمفية (*AlAlmfyp*).

We evaluated the results of AMIRA's tokenization result in terms of three measures, precision, recall and F-measure using the following equations:

$$Precision = \frac{the\ number\ of\ words\ which\ have\ been\ tokenised\ correctly}{the\ number\ of\ words\ which\ have\ tokenised}$$

$$Recall = \frac{the\ number\ of\ words\ which\ have\ been\ tokenised\ correctly}{the\ number\ of\ words\ which\ need\ to\ be\ tokenised}$$

$$F - measure = \frac{2 * recall * precision}{recall + precision}$$

The AMIRA tool has achieved 91.22%, 87.15% and 89.13% for precision, recall and F-measure, respectively. Two categories of errors are identified:

- False positive errors that occur when AMIRA tokenises a word that does not need to be tokenised.
- False negative errors that occur when AMIRA misses tokenising word that needs to be tokenised.

One of the most common false positive errors was tokenising words where the first letter after the ال (Al) determiner is ل (L). Examples of these words are: اللعابية (*AllEAbyp* - salivary), اللمفية (*Allmfyp* - lymphatic), اللوزتين (*Allwztyn* - tonsils) and اللوكيميا (*AllwkymyA* - leukemia). Some of these errors may be related to the limited data set used by AMIRA's classifier. These errors were corrected manually before moving to the next task. AMIRA adds a ا (A) letter after the determiner in these words so the wrong results of tokenising these words are الالعابية (*AlAlEAbyp*), الالمفية (*AlAlEAbyp*), الالوزتين (*AlAlwztyn*), and الالوكيميا (*AlAlwkymyA*). A proposed solution for this error is not to tokenise any words that have a double letter ل (L), unless the double ل (L) is the first two letters, or to insert a good number of examples of these words into the training data if the tokenisation system is using a machine learning technique, as with AMIRA. Regarding false negative errors, the main words were those that started with the ب (b) preposition. Examples of these words are: بالسرطان (*bAlsrTAn* -by cancer), بحسب (*bHsb* - according to), بالدهون (*bAldhwn* - with fats), باليود (*bAlywd* - with iodine). It is possible to split the ب (b) preposition if the following letters are the determiner ال (Al). This is because Arabic words which start with بالـ (bAl), where the ب (b) is an original letter of the word, are very uncommon. In order to examine how common these words are, the ANERcorp corpus, which consists of around 150,000 tokens (Benajiba et al., 2007) was used. Among the ANERcorp, 1104 words start with بالـ (bAl). However, in only 21 of these is بالـ (bAl) part of the original word, and nine words of the 21 words are actually non-Arabic. The rest of the words are a repetition of only four Arabic words which are بالغة (*bAlghp* - exaggerate), بالغ (*bAlgh* - adult), بال (*bAl* - shabby) and بالي (*bAly* - shabby). Creating a gazetteer for words which start with بـالـ (bAl) when the ب (b) is an original part of the word, would assist the tokenisation of such words.

4.3 POS Tagging

AMIRA is also applied to perform POS tagging. Three different tag sets are available: Bies tag set, Extended Reduced tag set (ERTS) and Extended Reduced tag set + person information (ERTS_PER). The Bies tag set was developed by Ann Bies and Dan Bikel and consists of 24 tags. It ignores certain Arabic distinctions, for example, it treats the dual form, a common form in Arabic language, as a plural. It also can not specify gender in both verbs and nouns. The ERTS tag set has 72 tags and provides additional morphological features to the Bies tag set, and can handle number (singular/dual/plural), gender (feminine/masculine) and definiteness (the existence of the definite article or not). In addition to the tags in the ERTS tag set, the

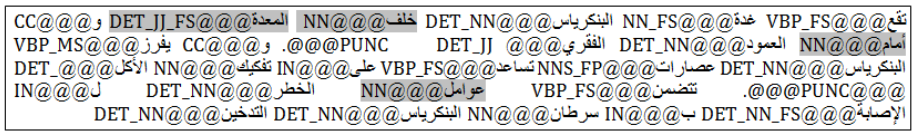


Fig. 3. A sample of the POS tagging task result

ERTS_PER specifies the use of the first, second and third person voice. The ERTS, which was selected for the POS tagging task, has many relevant morphological features to our corpus while Person information is a less important feature as our data only has the third person voice. Figure 3 displays a sample of the POS tagging task result.

In the above example, AMIRA assigned a noun tag to the place adverbs خلف (behind) and أمام (in front of). It also assigned an adjective tag to the genitive noun المعدة (stomach). Amira also failed in assigning a plural noun tag NNS to the word عوامل (factors). We evaluated the results of AMIRA's POS tagging in terms of the accuracy. POS tagger accuracy is the number of correctly tagged tokens divided by the total number of tokens. AMIRA achieved an accuracy of 84.09%. However, Arabic POS taggers still need more research efforts to improve the accuracy and reach a standard equal to Stanford POS tagger for English language which has achieved 97.3% accuracy (Manning, 2011). The areas where AMIRA performed less than the average is explained below.

- **Broken plurals**

Arabic has three types of plurals: the broken plural, the sound masculine plural and the sound feminine plural. The most used type is the broken (irregular) plural, constituting about half of all plurals in Arabic (Habash, 2010). AMIRA has limited capability to assign an appropriate POS tag to broken plurals, as 32.02% of AMIRA errors are related to broken plural words. For instance, AMIRA assigns a singular feminine word tag (DET_NN_FS) to the broken plural words الأوعية (utensils), الأنسجة (tissues) and الأقنية (ducts). It also failed to assign a plural noun tag (NNS) to most of the other broken plural words. Examples of these words are الأطباء (doctors), سُبُل (ways) and خلايا (cells). Broken plurals can be formed using more than 20

morphological patterns. Furthermore, an Arabic word might have more than one plural. For instance, the word أسد (lion) has five different broken plural forms (آساد - أسود - أسُد – أسَدة – أسْد). Therefore, it can be quite difficult to identify a solution for broken plural POS tagging. We propose to improve the performance of broken plurals POS tagging by using machine learning classifier techniques such as neural networks, or decision tree. In the literature, Goweder et al (2004) examine different methods in order to identify the broken plural. Then concluded that the dictionary and decision tree methods achieved the highest results in identifying broken plurals.

- **Adverbs**

In Arabic, there are two main types of adverb: those describing time and others referring to place or location. AMIRA assigned a noun tag (NN) to most adverbs in our corpus. Examples of these adverbs are: خلف (behind), أسفل (at the bottom of) and بعد (after). We propose to create an adverb gazetteer and use it as a binary feature to feed the machine learning classifier.

- **Adjective and genitive nouns**

One of the most frequent errors in AMIRA's POS output is assigning an adjective tag (JJ) to genitive nouns (المضاف إليه). For instance, AMIRA assigns a JJ tag to the word 'stomach' in the phrase سرطان المعدة (cancer of the stomach), the word 'patient' in the phrase فرصة المريضة (the patient's chance) and the word 'appetite' in the phrase نقصان الشهية (loss of appetite). There are some grammatical differences between adjectives and genitive nouns, in Arabic grammar. Adjectives and the nouns that they modify must agree in number (singular/dual/plural), mood (indicative/subjunctive/ genitive) and in indefiniteness and definiteness (presence of the definite article). In the above examples, the adjectives and the nouns that they modify disagree in both mood and the indefiniteness and definiteness. Using these grammatical differences as features in the data training phase will improve the task of differentiation between adjectives and genitive nouns.

5 Conclusion

Tokenisation and POS tagging are two important tasks used at early stages of named entity recognition systems. Whilst these tasks may be seem less challenging when processing English texts, many challenges face their implementation for Arabic texts because of the complex morphological structure of the Arabic language. This paper has described some of these challenges encountered by the use of AMIRA to tokenise and POS tag articles related to cancer extracted from the health encyclopedia. The AMIRA tokeniser has achieved 91.22%, 87.15% and 89.13% for precision, recall and F-measure, respectively, while AMIRA POS tagger achieved 84.09% accuracy. The most common errors in the tokeniser output were in the words where the first letter after the ال (Al) determiner is ل (L). With respect to the POS tagging, the areas where AMIRA underperformed include broken plurals, adverbs,

adjectives and genitive nouns. Some of these errors can be addressed using machine learning techniques which will be the subject for future work.

Acknowledgment

This research is supported by Aljouf University, Saudi Arabia and Staffordshire University, UK.

References

Alkharashi, I. (2009) Person named entity generation and recognition for Arabic language. In: *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, Cairo, pp.205–208.

Al-Shalabi, R., Kanaan, G., & Al-Serhan, H. (2003*). New approach for extracting Arabic roots*. Paper presented at the International Arab Conference on Information Technology (ACIT'2003), Alexandria, Egypt.

Alsughayr A. (2013) King Abdullah Bin Abdulaziz Arabic health encyclopedia (www.kaahe.org): A reliable source for health information in Arabic in the internet. Saudi J Med Med Sci; 1: 53-4

Benajiba, Y., & Paolo, R. (2007) ANERsys 2.0: Conquering the NER task for the Arabic language by combining the maximum entropy with POS-tag information. In: *Proceedings of Workshop on Natural Language-Independent Engineering, 3rd Indian International Conference on Artificial Intelligence (IICAI-2007)*, Mumbay, pp.1814–1823.

Buckwalter T. (2002) *Buckwalter Arabic Morphological Analyzer Version 1.0* Linguistic Data Consortium, University of Pennsylvania.

Buckwalter, T. (2004). *Buckwalter Arabic morphological analyzer (BAMA) version 2.0.* linguistic data consortium (LDC) catalogue number LDC2004L02. ISBN1-58563-324-0.

Diab, M. (2009) Second Generation Tools (AMIRA 2.0): Fast and Robust Tokenization, POS tagging and Base Phrase Chunking. *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009.

Diab, M, Hacioglu, K., & Jurafsky, D. (2007) Arabic Computational Morphology: Knowledge-based and Empirical Methods, chapter Automated Methods for Processing Arabic Text: *From Tokenization to Base Phrase Chunking.Kluwer*/springer edition

Farghaly, A., & Shaalan, K. (2009) Arabic natural language processing: Challenges and solutions. *ACM Transactions on Asian Language Information Processing (TALIP),* pp.1–22.

Goweder, A., Poesio, M., De Roeck, A. N., & Reynolds, J. (2004). Identifying Broken Plurals in Unvowelised Arabic Tex. In *EMNLP* (pp. 246-253).

Grishman, R., & Sundheim, B. (1996). Message Understanding Conference-6: A Brief History. *In COLING* (Vol. 96, pp. 466-471).

Habash N. (2010) *Introduction to Arabic Natural Language Processing*. Synthesis Lecture on Human Language Technologies. A Publication in the Morgan & Claypool Publishers series, UAS.

Habash, N., Rambow, O., & Roth, R. (2009) MADA+TOKAN: A toolkit for Arabic tokenization diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR), Cairo, Egypt

Khoja, S. (1999) *Stemming Arabic Text*. Computing Department, Lancaster University, Lancaster, U.K

Larkey, S., & Connell, E. (2001) Arabic Information Retrieval at *UMass In TREC-10, The Tenth Text Retrieval Conference, TREC 2001*. Gaithersburg: NIST, 562-570

Manning, C. D. (2011). Part-of-speech tagging from 97% to 100%: is it time for some linguistics?. *In Computational Linguistics and Intelligent Text Processing* (pp. 171-189). Springer Berlin Heidelberg.

Mohamed, E., & Kübler, S. (2010) Arabic Part of Speech Tagging. *In Proceedings of the Seventh International Conference on Language Resources and Evaluation (LRE 2010C)*, 19-21 May, Valletta, Malta.

Peng, F., Feng, F., & McCallum, A. (2004). Chinese segmentation and new word detection using conditional random fields. *In Proceedings of the 20th international conference on Computational Linguistics* (p. 562). Association for Computational Linguistics.

Sawalha, M. & Atwell, E. (2009) Linguistically Informed and Corpus Informed Morphological Analysis of Arabic. In: *Proceedings of the 5th International Corpus Linguistics Conference CL2009*, 20-23 July 2009, Liverpool, UK.

Sawalha, M., & Atwell, E. (2008). Comparative evaluation of arabic language morphological analysers and stemmers. *In Proceedings of COLING 2008 22nd International Conference on Comptational Linguistics (Poster Volume))* (pp. 107-110). Coling 2008 Organizing Committee.

Voutilainen, A. (2003) Part-of-speech tagging. In R. Mitkov, editor, *The Oxford handbook of computational linguistics*. University Press, Oxford, pp. 219–232.

A Novel Stimulus and Analysis System for Studying the Neural Mechanisms of Natural Language Processing in the Human Brain

Alyssa Hwang[1] and Sara Chung[1]

[1]Department of Neurology, New York University
Hwangalyssa16@gmail.com
Sarac234@gmail.com

Abstract. Traditional experiments on language perception in the human brain used tightly controlled sets of stimuli, including short phrases and repetitions of words to analyze the effects on neurological recordings. However, critics argue that simple language stimuli outside the context of a conversation fail to capture the full spectrum of linguistic complexity during natural speech. We present a novel experimental strategy to study natural language use in epileptic patients undergoing electrocorticography while watching a Hollywood movie that contains many instances of natural interpersonal speech. Detailed analyses were created based on the language components available, involving multisensory speech and linguistic parsing of human language based on phonology, lexical access, semantics, and syntax. This experimental system promotes a more comprehensive understanding of the neural implementation of speech by allowing a shift from studying language in confined experimental conditions and introducing innovative approaches to studying the brain's capacity for understanding multimodal and naturalistic language.

1 Introduction

Remember that time you tried to understand a thick foreign accent? You might have asked the speaker to repeat the sentence, speak more slowly, or enunciate each syllable more clearly. You might have watched the speaker's lips or looked for gestures. This example illustrates the basic principles of communication: successful speech perception encompasses the integration of visual and auditory cues. Auditory cues are the more obvious stimuli related to speech, with one prime example being the sound of the speaker's voice. Visual cues consist primarily of the motions of the lower face, particularly the movements of the speaker's lips. While auditory processing plays a dominant role in understanding speech, the visual aspect also significantly affects speech comprehension (Ross et al. 2007). The McGurk Effect shows powerful evidence of this phenomenon: when the syllable /da/ is heard while the lip movement for the syllable [ga] is seen, the syllable /ba/ is perceived (McGurk & MacDonald 1976). This is done by presenting a video in which the speaker says the syllable [ga] accompanied with audio for the syllable /da/.

In previous experiments, one variable would be manipulated while everything else was held constant. The videos in these experiments often presented the image of just the lower face accompanied by the sound of a single word or a syllable. A typical experimental video would be one in which the speaker's lower face was shown repeating the word "house" three times. Although these experiments led to discoveries about multisensory integration in the brain, they were not reflective of naturalistic speech perception – a typical speech rate elicits up to ten words per second, with each word made up of several syllables. Furthermore, these preliminary experiments show little progress towards developing a method to monitor free behavior. Monitoring free behavior would be the most effective way to collect data since the recordings would not be altered due to experimental limitations. However, studying natural behavior in a free environment is especially difficult considering the multitudes of inconsistent environmental factors that obstruct quality experimental control (Dastjerdi et al. 2013).

Newer research has begun to find the specific time and location of neurological integration (Schepers 2014). Early studies have suggested that audiovisual integration occurs as a late phenomenon, taking place after early separate analyses of the auditory and visual unimodal signals. However, the most recent studies suggest that there may be some early bimodal integration of these speech components. Further supporting this claim, a recently published investigation of numerical processing in the parietal cortex using electrocorticography and video recording revealed that areas of the brain responsive to numerical words in controlled experiments are also responsive in free behavior (Dastjerdi et al. 2013). Throughout this project, we endeavored to develop methods that can allow us to answer questions such as: Are neural representations of speech co-active in controlled experiments and free behavior? Are these experiments indicative of social interactions? These types of research questions are novel and can only be developed with the use of methodology like the one we describe here.

Such experiments were made possible by the improvement of brain imaging technology, such as functional magnetic resonance imaging (fMRI), positron emission tomography (PET), and electrocorticography (ECoG). ECoG, also known as intracranial electroencephalography or iEEG, is the most important equipment for our study because it provides recordings directly from the brain surface, providing high spatial (mm) and temporal (ms) resolution. One defining feature of ECoG is the high signal to noise ratio by location of the electrodes that are directly on the cortex the brain. As a result, wide frequency ranges are recorded using this method and have shown to be more robust relative to less-invasive imaging methods such as fMRI and EEG.

While recording data from our subjects, high gamma band power was used as an indicator of neuronal activity. High gamma waves are within 80-200 hertz, and any oscillation that completes a phase in less than one second represents a high gamma band. These frequencies correlate and reflect the firing of neuronal populations close to the site of recording (UC Berkeley News). Recordings of lower frequency, 1-40 hertz, demonstrate local field potential and are less spatially selective but can be recorded from outside the head. High gamma waves are described in contrast to recordings of lower frequencies to show a tight relationship to increase in frequency and lower amplitude (UC Berkeley News). Because high gamma waves are very spatially selective, a time-frequency plot can be created to see the hot spots caused by a stimu-

lus in a specific region of the brain. This is a very reliable way of seeing which areas of the brain are activated when subject to certain stimuli.

This project can be seen as an intermediate step between traditional controlled experiments and free behavior monitoring. By using a movie to mimic normal conversation and a natural environment, we were able to collect data that represented a more natural setting while closely monitoring and controlling the testing environment and presentation. While the incidents of a natural environment are difficult to predict and analyze, the usage of a movie simulator offers the opportunity to be manipulated, prepared, and analyzed ahead of time while containing many aspects of natural environment, such as conversation, background noises, and music. The movie represents a reality we can control.

2 Methods

We propose an innovative system for studying the neural correlates of natural language processing by presenting and analyzing a complex stimulus set based on an auditory-visual movie within the confines of a hospital setting. The movie file has been manipulated to allow synchronization with the neural recordings. Post-hoc analyses include, but are not limited to, investigations of auditory-visual speech integration and basic linguistic properties such as phonemes.

2.1 Movie Selection

The movie *Zoolander,* produced by Village Roadshow Pictures and VH1 Films and running for 01:29:09 (hh:mm:ss) at 29 frames per second, was utilized as the continuous auditory-visual stimulus throughout this project. It was formatted as an MPEG-4 file. This humorous movie was chosen because of its variety of personalities, diversity of audio, visual, and audiovisual occurrences, and interactions between characters.

2.2 Synchronization with Neural Data

In order to synchronize the movie to the electrocorticography (ECoG; also intracranial electroencephalography, iEEG) data to allow for temporal alignment of the audio and visual input with the neural output, several photodiodes were embedded throughout the film. Photodiodes are white dots accompanied by a short tone. Five photodiodes an average of 48.5 milliseconds apart mark the beginning of the movie, and ten photodiodes an average of 49.33 milliseconds apart indicate the end. 67 other photodiodes are interspersed randomly throughout the movie, appearing approximately once per minute. The tones that accompany the white dots will be recorded by the clinical system. The patient does not hear the tones because the audio is split: the right channel plays the audio of the movie and the left channel plays the tones for synchronization. These channels are analogous to earphone wires: the patient hears sounds from the audio playing from the right “ear” while the audio from the left “ear” is connected to the clinical system. The photodiodes are viewed by a sensor which sends triggers to the clinical system that also records the iEEG data.

2.3 Audiovisual Video Annotation

The conditions “audio only” and “audiovisual” were naturally occurring throughout the movie. To simulate the condition “visual only,” three seconds of audio were removed from the movie at forty separate instances. The audio sections that were removed were spaced out equally throughout the film and only contained dialogue that is not essential to the plot of the movie to avoid obfuscating the higher-level comprehension of the movie’s elements, and thereby confusing the patient.

We annotated the movie to find when each segment occurs in the movie by dividing each instance of speech into one of the three conditions: audio only, visual only, and audiovisual. We were able to do this by utilizing the software ELAN, which allows the user to create annotations of media files in levels of representation, visualized as tiers. The program was created by Max Planck Institute for Psycholinguistics, The Language Archive, Nijmegen, The Netherlands. URL: http://tla.mpi.nl/tools/tla-tools/elan/. The video file together with the audio file was uploaded into ELAN in order to form the annotations.

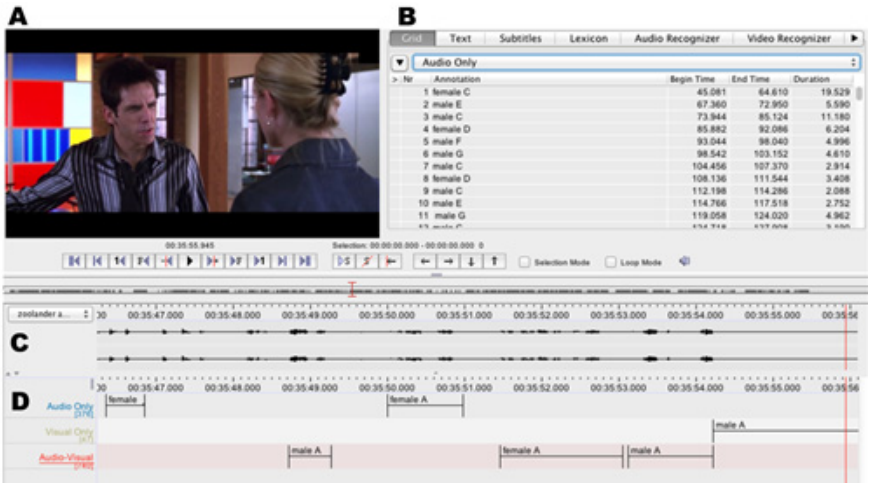


Fig. 1. ELAN software displays the visual content of the movie (A), the three tiers with indicated segments (B), a spectrogram for sounds and audio content (C), and the alignment of segments in relation to other tiers (D)

To mark the time segments in each condition, each instance of speech was manually tagged and organized into one of the tiers. A and AV time segments were marked at the start of the auditory signal while V time segments were noted with the onset of facial movement. The audio wave file was enlarged to 500% in ELAN for accuracy while we were listening to the segments. A new segment was started whenever speech paused for longer than two seconds or the speaker changed. The segments were between 0.084 seconds and 30.027 seconds long (average: 5.928s). Within the tiers marked A, V, or AV, individual identities of the speakers were coded and noted as

Female A, Male A, Female B, Male B, etc. Throughout the entire movie, there were 376 A segments, 47 V segments, and 740 AV segments, each to be synchronized to epoch durations in the neural data.

2.4 Linguistic Analysis

The dialogue of the movie was transcribed by a combination of listening to the audio and comparing notes with different sets of subtitles and scripts available online. After transcribing the speech of the movie, the appropriate phonemes were found to match up with the transcript using an automated process. Phonemes systematically represent different sounds in human language. After transcribing the movie, the dialogue was separated into their A, V, or AV files to match the annotated segments. The movie was divided into three sets of eight roughly even sections, A, V, and AV, to keep the file sizes lower, more manageable, and organized.

FAVE-align, an automatic alignment program made by the University of Pennsylvania Linguistics Lab, was used to align the transcript and phonemes of the segments with the audio of the movie. Some words, such as "Zoolander,"not included in the algorithm's dictionary were added to it with the appropriate accompanying phonemic transcription.

Finding the phonemic spelling for such unrecognized words was possible using the same FAVE-align program with the option "Check transcription for unknown words." The phonemes were generated by the CMU Pronouncing Dictionary (http://www.speech.cs.cmu.edu/cgi-bin/cmudict**)** with the option "show lexical stress." Fragments, truncated words such as "Zoolan-," were not recognized by FAVE-align either. To resolve this issue, each fragment was given a code (F1, F2, F3, etc …) and replaced by its code in the transcript. The code was then entered into the dictionary file with its appropriate phoneme translation. The input for this software was the TextEdit file for each segment, the .wav file containing its audio, and the master dictionary TextEdit file. This returned the output: a TextGrid file indicating the start and end times of each word and phoneme organized by character. This resulting TextGrid file was used as the input for a MATLAB script. The data concerning the phonemes and their time lapses was extracted and represented in a simpler form by an Excel spreadsheet. After receiving the TextGrid files, each was manually checked to remove words that were misaligned with the audio.

Praat is a software that displays the audio wave, spectrogram, and TextGrid alignment when the TextGrid and its correlating audio file are uploaded. To check if the transcript and audio were lined up correctly, we listened to each individual word. If a word was incorrect we removed it, replacing the word and its phonemes in the TextGrid with "RM." This reinforced the validity of the FAVE-align algorithm. This was done for all components of the movie (audio only, visual only, and audiovisual). In total, there were 1,163 segments. The audio section had 376 segments, the visual section had 47, and the audiovisual section had 740.

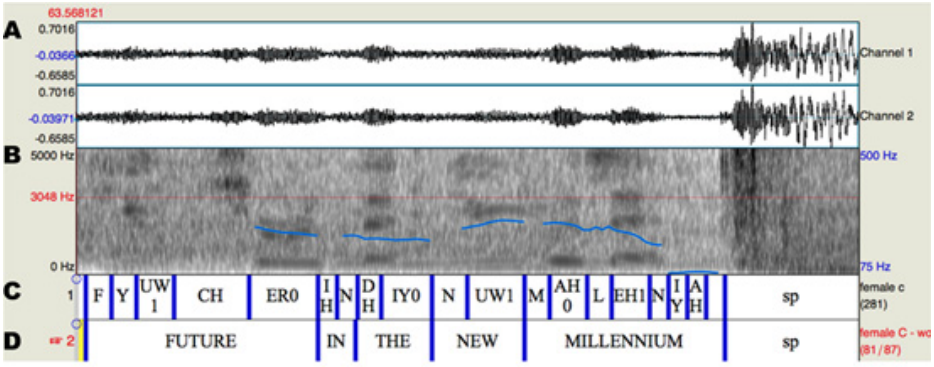


Fig. 2. Praat displays a periodogram for audio content (A), a spectrogram of sound waveforms (B), and the alignment of individual phonemes (C), as well as individual words and spaces (D)

3 Discussion

Here we present a novel procedure for studying the mechanics of natural language processing. We have engineered a solution that allows the presentation of more natural speech stimuli to patients with brain electrode implants, their synchronization with the neural iEEG recordings, and an in-depth classification of the auditory and visual speech stimuli. This will allow future detailed and varied neurophysiological analysis of free behavior language processing in the brain. The design and use of such a set-up has not yet been reported in scientific literature and represents a step towards understanding how the brain processes language outside tightly controlled laboratory conditions. The Society for Neurobiology of Language has recently organized at its annual meeting a novel symposium entitled "A Neurobiology of Natural Language Use?' (Society for the Neurobiology of Language) highlighting the novelty, relevance, and interest by the scientific community in studying natural language. While recent methodological developments in brain imaging techniques, such as functional magnetic resonance imaging (fMRI), have made the study of natural language more feasible, our method is the first to allow the use of powerful electrocorticography recordings from brain implants in conscious and performing humans to predict the effects of natural speech. The design and specific analysis of the stimulus set presented here, together with the high trial count (in the current case, 1163 "trials") afforded by intracranial brain recordings, enable a neuroscience investigator to ask an almost unlimited number of research questions related to language processing.

The parsing of the audio and video stimuli into linguistic segments at the sentence, word, and phoneme level allows linguists to explore a great deal in regards to, but not limited to, auditory-visual speech integration, phoneme-specific localization, lexical access, semantics, and syntax. Careful research and application of the method we created can lead to the discovery of specific locations of language processing in the brain and more accurate interpretations of neuronal signals than in previous studies. Our method could be further refined and modified to study free behavior in a natural setting using the principles of neurocinematics. In the seminal study of neurocinemat-

ics, experimental data indicated that a group of people reacted similarly when watching the same movie, and that the movie could "control" the viewers' neural responses (Hasson et al. 2008). This idea could be applied to the study of free behavior - people's reactions and changes in brain activity to the same situation could be observed. Current research already indicates that certain brain activity in the superior temporal gyrus correlates to the phonetic features of the English language (Mesgarani et al. 2014). Clearer understanding of linguistic processes in the human brain could be used to create technology that predicts the words being thought. Further developments could then be used to advance brain-computer interface technology. This technology would be of substantial use and benefit for those with severe motor disabilities. Technology that can predict a person's thoughts and translate neuronal signals into movement would revolutionize the creation of Assistive Augmentative Communication devices (AAC).

4 Acknowledgments

We thank Thomas Thesen, Paul Del Prato, Megan Marshall, and the rest of the Department of Neurology at New York University for their insight, advice, and assistance throughout the development of this paper.

References

Altieri, Nicholas. "Audiovisual Integration: An Introduction to Behavioral and Neuro cognitive Methods." *US National Library of Medicine National Institutes of Health*. N.p., n.d. Web. 16 Sept. 2014. <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3775457/>.

Brill, Eric. *Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging*. N.p.: n.p., n.d. *Association for Computational Linguistics*. Web. 22 Sept. 2014. <http://www.aclweb.org/anthology/J95-4004>.

Calvert, Gemma A., and Thomas Thesen. "Multisensory Integration: Methodological Approaches and Emerging Principles in the Human Brain." *Journal of Physiology - Paris* (2004): n. pag. Print.

Dastjerdi, Mohammad, et al. "Numerical Processing in the Human Parietal Cortex during Experimental and Natural Conditions." *Nature Communications* (2013): n. pag. Print.

Hasson, Uri, et al. *Neurocinematics: The Neuroscience of Film*. N.p.: n.p., 2008. Print.

Hill, NJ, et al. *Recording Human Electrocorticographic (ECoG) Signals for Neuroscientific Research and Real-Time Functional Cortical Graphing*. N.p.: n.p., n.d. *National Center for Biotechnology Information*. Web. 22 Sept. 2014. <http://www.ncbi.nlm.nih.gov/pubmed/22782131>.

McGurk, H., & McDonald, J. (1976). Hearing lips and seeing voices. Nature, 264, 746–747.

Mesgarani, N., et al. *Phonetic Feature Encoding in Human Superior Temporal Gyrus*. N.p.: n.p., 2014. *National Center for Biotechnology Information*. Web. 25 Sept. 2014. <http://www.ncbi.nlm.nih.gov/pubmed/24482117>.

Navarra, Jordi, et al. "Multisensory Interactions in Speech Perception." *The New Handbook of Multisensory Processing*. Ed. Barry E. Stein. N.p.: n.p., 2012. N. pag. Print.

Reale, R. A., et al. *Auditory-Visual Processing Represented in the Human Superior Temporal Gyrus*. N.p.: n.p., n.d. *NYU Langone Medical Center*. Web. 16 Sept. 2014. <http://www.med.nyu.edu/thesenlab/wp-content/uploads/2014/04/reale_etal_neurosc_2007.pdf>.

Ross, Lars A., Dave Saint-Amour, Victoria M. Leavitt, Daniel C. Javitt, and John J. Foxe. "Do You See What I Am Saying? Exploring Visual Enhancement of Speech Comprehension in Noisy Environments." *Cerebral Cortex* 17 (2007): 1147-153. Print.

Sanders, Robert. "Slow Brain Waves Play Key Role in Coordinating Complex Activity." *UC Berkeley News*. N.p., n.d. Web. 22 Sept. 2014. <http://berkeley.edu/news/media/releases/2006/09/14_theta.shtml>.

Schepers, Inga M., Daniel Yoshor, and Daniel S. Beauchamp. *Electrocorticography Reveals Enhanced Visual Cortex Responses to Visual Speech*. N.p.: n.p., 2014. Print.

Sinclair, John M. "The Automatic Analysis of Corpora." *Directions in Corpus Linguistics*. By Mouton De Gruyter. Ed. Jan Svartvik. N.p.: n.p., n.d. *Google Books*. Web. 22 Sept. 2014.

Society for the Neurobiology of Language. N.p., n.d. Web. 25 Sept. 2014. <http://www.neurolang.org/symposium/>.

The Selection of Classifiers for a Data-driven Parser

Sardar Jaf[1], Allan Ramsay[1]

[1]The University of Manchester, Faculty of Engineering and Physical Sciences, School of Computer Science, Manchester, United Kingdom
{sardar.jaf, allan.ramsay}@manchester.ac.uk

Abstract. There is a large number of classifiers that can be used for generating a parse model; i.e., as an oracle for guiding data-driven parsers when parsing natural languages. In this paper we present a general and simple approach for generating a parse model. Additionally, we present a large number of experiments on various classifiers. We also present the effect of various parse models, which are generated from different classifiers, on a data-driven parser to see the way each model contributes to parsing performance.

1 Introduction

The objective of this study is to present an approach for generating different parse models, which are used for guiding parsers during natural language parsing, from different machine learning classifiers. There are various classification algorithms that can be used for this purpose. However, different classifiers may learn from a set of data differently, which means that they may affect parsing performance in different ways. In Section 2 we present a data-driven parser that we have used for examining the effectiveness of different parse models, which are generated from different classifiers. In Section 3 we show a simple approach for generating a parse model from the J48 classifier while in Section 5 we show the accuracy of a large number of classifiers. Section 6 covers the effect of each parse model on parsing performance. Finally, in Section 7 we compare our parser with the arc-standard algorithm of MaltParser.

2 A Data-driven Shift-Reduce Parser

Our parser is based on the arc-standard algorithm of MaltParser (Kuhlmann and Nivre, 2010). This algorithm deterministically generates dependency trees using two data-structures: a queue of input words, and a stack of items that have been looked at by the parser. Three parse actions are applied to the queue and stack: SHIFT, LEFT-ARC and RIGHT-ARC (we will write LA and RA for LEFT-ARC and RIGHT-ARC respectively to save space). SHIFT moves the head of the queue onto the top of the stack, LA makes the head of the queue a parent of the topmost item on the stack and pops this item from the stack, and RA makes the topmost item on the stack a parent of the head of the queue; RA removes the head of the queue and moves the topmost item on the stack back to the queue. MaltParser uses a support vector machine classifier for

generating a parse model from a set of parsed trees, which is used for predicting the next parse action given the current state of the parser.

We will call our parser NDParser. At each parse step, we generate a state for LA, RA, and SHIFT, and we will assign different scores to each state. For example, a score is computed for each newly generated state by computing two different scores: (i) a score that is based on the recommendation made by a parse model. For instance, when generating a SHIFT state the parser gives a score of 1 if a SHIFT operation is recommended by the model. Otherwise a score of 0 is given (and the same applies to LA and RA). (ii) The score of the state that the new state is derived from. The sum of these two scores is then assigned to the newly generated state. The advantage of assigning a score to a parse state is that we can rank a collection of parse states by using their scores and then process the state with the highest score. In order to efficiently process a potentially large set of states, we use dynamic programming for ranking competing states with respect to their plausibility (the plausibility of a state is based on its score.) The ranked states are then stored in a chart table (Kay, 1973) and the most plausible state is explored by the parser, where new states are generated by using SHIFT, LA, and LR operations. This way we combine features of chart parsing with shift-reduce parsing.

3 The Generation of a Parse Model

The effectiveness of a parse model largely depends on the classifier's ability to correctly classify a Treebank. The Penn Arabic Treebank (Maamouri and Bies, 2004), which we converted it to Dependency format, was used for parser training and testing. We have experimented with several classifiers that are available in the `WEKA' toolkit (Hall et al., 2009) for classifying a set of training data. The output of each classifier is then used for generating a parse model, which is then used for examining the effect of each model on parsing performance. The following steps explain a general and simple approach for generating a parse model from a dependency Treebank:

Step 1. Forced parsing: we use a shift-reduce parser for parsing the training data, which contains the parsed trees of each sentence. The parsed tree of each sentence are used as a grammar to parse the sentence, which is used as a guide to parse the sentence and record parse states during training.

Step 2. Collecting parse states: during training we obtain a set of parse states, i.e., state:action pairs where the condition is the state of the queue and stack (i.e., the items on the queue and stack) and the action is the parse operation that the parser performed (which is either SHIFT, LA, or RA). Consider, for instance, the parsed tree in Fig. 1 for the sentence '*the cat sat on the mat*'.

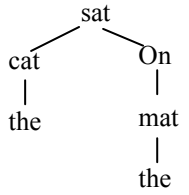


Fig. 1. Dependency tree for the sentence '*the cat sat on the mat'*

Fig. 2 shows the transitions that the parser uses for producing the tree for the sentence '*the cat sat on the mat'*. Note that whilst constructing the training data we will not perform any LA or RA operations if a dependency daughter is the head of an item that is not inspected yet, as illustrated from step 6 of Fig. 2 where we perform SHIFT instead of RA, since performing RA at this point would make it impossible to later make *on* the head of *mat* because RA would remove *on* from the queue, which would prevent it from becoming the head of *mat* which is still on the queue.

Dependency relations: (sat>cat) (sat>on) (cat>the) (on>mat) (mat>the)

Steps	Action	Queue	Stack	Arcs
1	-	[the,cat,sat,on,the,mat]	[]	-
2	SHIFT	[cat,sat,on,the,mat]	[the]	-
3	LA	[cat,sat,on,the,mat]	[]	A1=(cat>the)
4	SHIFT	[sat,on,the,mat]	[cat]	A1
5	LA	[sat,on,the,mat]	[]	A2=A1+(sat>cat)
6	**SHIFT**	**[on,the,mat]**	**[sat]**	**A2**
7	SHIFT	[the,mat]	[on,sat]	A2

Fig. 2. Parse states when parsing the sentence '*the cat sat on the mat*'

We can treat the sequences shown in Fig. 2 as a set of data-points which indicate what the parser should do in a given state -- for instance, in a situation like in step 6 in Fig. 2 the parser should use SHIFT instead of RA for the reason explained above.

Given a set of such data-points, it is possible to extract and record the parse states and train a classifier for building a parse model, which can be used for predicting parse operation; i.e., it can be used for guiding the parser. The task here is to classify intermediate states of the parser into three groups: cases where SHIFT should be performed, cases where LA should be performed, and cases where RA should be performed.

Step 3. Preparing recorded parse states for classification: from the set of parse states that we obtained in step 2, we populate an *.arff* file with the correct data format, i.e., the format that is accepted by WEKA. An example of a set of WEKA-style data format is shown in Fig. 3, which is based on the parse states shown in Fig. 2. Here we have extracted the word forms as a feature for learning but it is possible to use a number of different features (such as POS tags, word position etc.) as values for the queue and the stack attribute parameters.

```
@relation states
@attribute queue_word_pos_1 {'the', 'cat', 'sat', 'on', 'mat', '-' }
@attribute queue_word_pos_2 {'cat', 'sat', 'on', 'the', 'mat', '-'}
@attribute stack_word_pos_1 {'-','the', 'cat', 'sat', 'on'}
@attribute stack_word_pos_2 {'-', 'sat', 'on'}
@attribute parse_action {'SHIFT', 'LEFT-ARC', 'RIGHT-ARC'}
@data
```

```
‘the’, ‘cat’, ‘-’, ‘-’, ‘SHIFT’
‘cat’, ‘sat’, ‘the’, ‘-’, ‘LEFT-ARC’
‘cat’, ‘sat', ‘-’,‘-’’SHIFT ’
‘sat’, ‘on’, ‘cat', ‘-’, ‘LEFT-ARC’
 …
```

Fig. 3. An example of data for an *.arff* file

Additionally, one can use many different window sizes for the queue and the stack in the data selection as instances for the classification algorithms to learn from. In Fig. 3. we use a window size of two items for the queue and two items for the stack, while the dash mark (‘-’) represents an empty item where the queue or the stack did not contain an item in the given position.

Step 4. Training a classifier using the *.arff* file: we supply WEKA with the data prepared in step 3 (i.e., the *.arff* file) and then we select a classification algorithm for learning. Fig. 4. is an example of the J48 classification algorithm output from WEKA.

Step 5. Generating a parse model from the classification output: finally, we convert the output produced by the classification algorithm to an appropriate state-action model, which is used for guiding the parser to parse new sentences. Fig. 5. is a sample of some states and actions we have extracted from the J48 (Quinlan, 1992) classifier's output.

```
Satck_word_pos_1 = ?: SHIFT (8430.0)
Stack_word_pos_1 = ABBREV
|    queue_word_pos_1 = ABBREV
|    |    queue_word_pos_2 = ?: RIGHT-ARC (6.0)
|    |    queue_word_pos_2 = ABBREV
|    |    |    queue_word_pos_3 = ?: RIGHT-ARC (5.0)
|    |    |    queue_word_pos_3 = ABBREV: RIGHT-ARC (2.0)
…
```

Fig. 4. An example of the J48 algorithm output using WEKA

```
states(
QUEUE,
STACK, [
word_pos(STACK, 1, ‘-’), ‘SHIFT’,
word_pos(STACK, 1, ‘ABBREV’), [
word_pos(QUEUE, 1, ‘ABBREV’), [
word_pos(QUEUE, 2, ‘-’), ‘RIGHT-ARC’,
… ] ] ] ] ] ).
```

Fig. 5. An example of a state-action model

4 Label Assignment to Dependency Relations

In this section we show the way we assign labels to dependency relations, which is largely different from the way this is implemented in the standard implementation of MaltParser. As in the arc-standard algorithm, for each dependency relation between two words, a label is attached to indicate the grammatical function of the daughter item with its parent. However, the way we assign labels to dependency relations during parsing is that we extract patterns from the training data during the training phase. This contrasts with the approach used in MaltParser whereby labels are predicted with the LA and RA actions of the parser which are learned during the training phase.

Each pattern consists of a dependency parent, a list of *n* part-of-speech (POS) tagged items, a dependency daughter, a label, and the frequency of the pattern in the training data. A schema of a pattern is shown in Fig. 6. The first element of the pattern is a parent item, the second is a list of up to *n* POS tagged items between a parent item and its daughter in the original text, the third is the daughter of a parent item, the fourth element is the label for the dependency relation and the last element is the frequency of the pattern recorded during training. Fig. 6. shows the pattern when PARENT is assigned as the parent of DAUGHTER where there are up to *n* POS tagged items between them then their dependency label is LABEL, and the last element indicates that the pattern occurred j times during training.

PARENT, [POS1,...,POS*n*], DAUGHTER, LABEL, j

Fig. 6. A schema of a pattern for a label

During the evaluation phrase, we show three different parsing accuracy measures, those are: (i) Labelled Attachment Scores (LAS), which is the percentage of the correct dependency relations with the correct labels of the dependency relations (DEPREL) between tokens; (ii) Unlabelled Attachment Score (UAS), which is the percentage of correct dependency relation (i.e., the percentage of tokens with correct heads) regardless of the DEPREL; and (iii) Labelled Scores (LS) which is the percentage of tokens with the correct dependency label.

5 Evaluating Different Classifiers

Table 1 contains the accuracy of various classifiers that were used for classifying the training data that we have mentioned in Step 2 of Section 3. We consider a classifier appropriate for producing a parse model if it meets two requirements: (i) it produces good classification accuracy. Although the accuracy of the classifiers that are presented in Table 1 may not directly reflect the accuracy of a parser that uses its recommendations but, a classifier that produces a high level of accuracy is more likely to assist a parser to make more informed parse decisions at each parse step than a classifier that produces a low level of accuracy; and (ii) its output can be used for generating a parse model which can be used for making recommendations to a data-

driven parser, for example, what action (SHIFT, LA, or RA) the parser should take in a specific situation. We have used various features for training different classification algorithms. These features included POS tags, word forms, word locations in sentences, their spans (i.e., their start and end positions in sentences). Additionally, we have used a combination of these features such as word forms with POS tags, word forms with word location or word spans, and similar combination of POS tags with other features. Also, various window sizes are used for the queue and stack, ranging between two items to four items. The use of these features for training each classifier along with the classification accuracy is presented in Table 1. Previous experiments by Jaf and Ramsay (2013) indicated that using a window size of more than four items on the queue or stack did not yield better results, hence we have used up to four items in this experiment.

Table 1. Classification accuracy with various feature and setting. *W* = Word, Loc = *Item location in sentence*, *POS = part-of-speech tags*, and *Span = start and end position of a word*

J48							
Items on Queue	2	3	3	3	4	4	4
Items on Stack	4	2	3	4	2	3	4
W (%)	68.24	68.29	68.37	68.53	68.56	68.67	68.81
W + Loc (%)	71.92	72.23	71.88	71.67	72.41	72.11	71.80
W + Loc + span (%)	71.73	72.76	72.25	72.00	72.87	72.45	72.17
W + span (%)	70.17	70.81	70.64	70.43	70.83	70.79	70.56
POS (%)	84.94	85.63	85.77	85.80	85.89	86.05	86.04
POS + Loc (%)	85.27	85.89	85.91	85.92	86.96	86.08	86.09
POS + Loc + span (%)	85.23	85.81	85.84	85.92	85.95	85.97	85.99
POS + span (%)	85.00	85.71	85.69	85.67	85.88	85.88	85.88
W + POS (%)	85.28	86.23	86.24	86.24	86.46	86.47	86.39
W + POS + Loc (%)	85.83	86.57	86.53	86.45	86.63	86.57	86.54
W + POS + Loc + span (%)	85.83	86.48	86.54	86.47	86.49	86.55	86.44
Word + POS + span (%)	85.79	86.50	86.45	86.43	86.53	86.49	86.36
LiBSVM							
Items on Queue	2	3	3	3	4	4	4
Items on Stack	4	2	3	4	2	3	4
W (%)	-	-	-	-	-	-	-
W + Loc(%)	-	-	-	-	-	-	-
W + Loc + span (%)	-	-	-	-	-	-	-
W + span (%)	-	-	-	-	-	-	-
POS (%)	74.62	75.41	75.43	75.39	75.62	75.63	75.55
POS + Loc (%)	-	-	-	-	-	-	-
POS + Loc + span (%)	-	-	-	-	-	-	-
POS + span (%)	-	-	-	-	-	-	-
W + POS (%)	-	-	-	-	-	-	-
W + POS + Loc (%)	-	-	-	-	-	-	-
W + POS + Loc + span (%)	-	-	-	-	-	-	-
W + POS + span (%)	-	-	-	-	-	-	-
Id3							
Items on Queue	2	3	3	3	4	4	4
Items on Stack	4	2	3	4	2	3	4
W (%)	67.65	67.94	67.77	67.68	67.97	67.79	67.62
W + Loc (%)	62.37	65.22	63.04	61.89	64.41	62.55	61.49

W + Loc + span (%)	62.69	64.85	63.18	62.26	64.23	62.79	62.00
W + span (%)	61.21	63.60	61.84	60.71	62.81	61.25	60.36
POS (%)	81.64	83.41	81.78	80.54	81.47	79.70	78.81
POS + Loc (%)	74.57	75.48	75.04	74.95	74.83	74.65	74.61
POS + Loc + span (%)	74.51	75.42	74.92	74.83	74.85	74.63	74.57
POS + span (%)	74.28	75.17	74.71	74.59	74.64	74.41	74.33
W + POS (%)	81.02	82.89	81.29	80.36	81.04	79.67	79.09
W + POS + Loc (%)	74.96	75.73	75.39	75.33	75.29	75.07	75.04
W + POS + Loc + span (%)	74.79	75.64	75.22	75.15	75.21	75.00	74.93
W + POS + span (%)	74.55	75.45	75.04	74.97	75.04	74.81	74.73
RandomTree							
Items on Queue	2	3	3	3	4	4	4
Items on Stack	4	2	3	4	2	3	4
W (%)	68.00	68.27	68.26	68.32	68.47	68.51	68.50
W + Loc (%)	70.25	70.64	69.35	68.67	70.19	69.36	69.83
W + Loc + span (%)	-	70.27	-	-	69.97	-	-
W + span (%)	67.46	68.99	68.25	67.39	68.67	67.79	67.89
POS (%)	83.71	85.28	84.71	84.26	84.78	84.31	83.69
POS + Loc (%)	79.18	81.41	80.15	78.84	80.09	78.18	80.12
POS + Loc + span (%)	76.32	79.41	78.02	76.26	78.79	77.42	76.47
POS + span (%)	79.19	80.89	78.69	77.78	79.93	77.28	76.95
W + POS (%)	83.62	85.37	84.34	83.57	84.46	83.33	83.28
W + POS + Loc (%)	80.10	81.84	80.62	79.17	80.27	79.03	77.99
W + POS + Loc + span (%)	77.3	79.59	77.50	76.33	78.09	76.77	75.02
W + POS + span (%)	78.42	80.58	77.87	77.34	78.95	76.87	77.17
NaiveBayes							
Items on Queue	2	3	3	3	4	4	4
Items on Stack	4	2	3	4	2	3	4
W (%)	60.13	65.95	65.48	63.37	65.06	64.68	64.60
W + Loc (%)	57.02	64.12	57.03	55.68	62.21	54.67	52.74
W + Loc + span (%)	49.98	47.29	44.51	47.60	45.28	42.79	45.28
W + span (%)	53.47	55.32	48.75	51.09	52.30	46.46	48.57
POS (%)	70.42	76.78	76.19	74.02	76.01	75.38	74.17
POS + Loc (%)	64.05	74.70	71.13	67.13	72.39	70.68	67.1
POS + Loc + span (%)	58.43	65.72	58.22	57.11	61.27	55.49	54.24
POS + span (%)	61.15	70.93	65.13	61.31	67.62	63.37	59.63
W + POS (%)	66.00	74.67	73.66	71.04	72.12	72.21	71.02
W + POS + Loc (%)	62.25	72.50	69.20	63.57	69.13	67.89	62.70
W + POS + Loc + span (%)	57.62	64.58	56.72	55.55	59.78	53.95	52.73
W + POS + span (%)	59.98	69.69	62.84	59.08	65.33	60.50	56.89

During the evaluation of the classifiers, some widely used classifiers did not yield encouraging results. For example, the LiBSVM classifier (Chang, 2001) which is used in MaltParser did not perform well with the set of features that we have supplied. It only managed to learn successfully from one feature (POS tags), while the accuracy was well below the accuracy of some of the other classifiers. The entries for LiBSVM in Table 1 are incomplete because training takes so long (3 days per case) that future experiments seemed infeasible. However, the fact that it produces no better classification than the J48 classifier in the cases that we have looked at suggests that it is unlikely to substantially outperform it in the remaining cases.

From the large number of experiments we have conducted on several classifiers, we will evaluate NDParser on them in the following section.

6 Evaluating NDParser with Various Classifiers

As presented in Table 2 the classification accuracy varies because each classifier learns differently from the set of training data. In this section, we investigate the effect of different classifiers on parsing. Our objective is to identify the classifiers that help the parser perform best in terms of accuracy and speed (We measure speed as the number of seconds per dependency relation). These experiments also highlight whether different parsing models, which are generated by using different classifiers, contribute in different ways to parsing performance. The optimal classification of accuracy may not necessarily lead to optimal parsing performance. Hence, it is necessary to investigate the effectiveness of different classifiers parsing performance.

Table 2. Parser evaluation with different classifiers, features and settings. *Q* = Queue size, *S* = Stack size, *POS* = part-of-speech tags, *RT* = RandomTree

Classifier	Features	Q	S	UAS (%)	LAS(%)	LS(%)	Spee
J48	**POS**	**4**	**3**	**74.5**	**71.0**	**93.6**	**0.081**
J48	POS	4	4	74.1	70.5	93.6	0.086
J48	POS + location	4	2	70.3	67.0	93.3	0.146
J48	POS + location + span	4	4	69.2	65.9	93.3	0.161
J48	POS + span	4	2	70.6	67.2	93.3	0.145
J48	POS + span	4	3	70.8	67.4	93.3	0.150
J48	POS + span	4	4	70.9	67.5	93.3	0.142
J48	Words + POS	4	3	71.4	67.9	93.5	0.096
J48	Words + POS + location	4	2	69.9	66.5	93.4	0.140
J48	Words + POS + location + span	4	3	68.0	64.8	93.1	0.183
J48	Words + POS + span	4	2	69.8	66.5	93.3	0.161
RT	POS	3	2	70.9	69.4	93.3	0.141
RT	POS + Location	2	1	68.6	65.5	92.9	0.154
RT	POS + Location + span	2	1	67.8	68.1	92.3	0.181
RT	POS + span	2	1	68.2	65.8	92.3	0.181
RT	Words + POS	2	2	70.0	66.6	92.3	0.196
RT	Words + POS + location	2	1	68.7	65.3	92.4	0.198
RT	Words + POS + location + span	2	1	66.8	63.6	92.1	0.196
RT	Words + POS + span	2	1	68.6	65.3	92.3	0.184
Id3	POS	3	1	70.6	67.2	93.4	0.083
Id3	Words + POS	2	2	68.1	64.8	93.3	0.099

From Table 1 we can identify the classification algorithms with the highest degree of accuracy. In this section, we trained NDParser using J48, RandomTree, and Id3 algorithms since they all classified the same set of training data with over 80% accuracy. For each of these algorithms we use the same settings that produced the optimal accuracy. For example, based on the results in Table 1 we will use the POS tags as a

training feature for the J48 algorithm with four items on the queue and three items on the stack because with this setting the algorithm produced 86.05% accuracy, while if we are using POS tags and their locations in a sentence as training features for J48 algorithm then we will use four items on the queue and two items on the stack because the algorithm performs best with this setting, which produced 86.96% accuracy.

The results of the evaluation of our parser are presented in Table 2. The best parsing performance is achieved when training the parser using the J48 classification algorithm on only POS tags as a feature and the window size for the queue and stack is four and three items respectively. The experiments in Table 1 show that training a classifier using a small set of features produces relatively similar classification accuracy to using a larger set of features. However, using a smaller set of features improves the parsing accuracy and speed.

7 A Comparison with the State-of-the-art Parser

In this section, we compare our parser with MaltParser, as shown in Table 3 where we have conducted a 5-fold cross validation on both parsers. We have trained our parser using the J48 classifier with POS tags as features and a window size of four items on the queue and three items on the stack. We can note that our parser is 43% more efficient than MaltParser. Although the unlabelled attachment score of our parser is slightly lower than that of MaltParser (0.7%), the labelled attachment score and the labelled accuracy is more accurate than MaltParser by 1% and 1.4% respectively. We believe that this improved accuracy of labelled attachment score and labelled score is because our parser have information about intermediate items between parent and daughter, which are collected during training (see Section 4 for more details) where such information is not available to MaltParser. MaltParser learns models that contain information about parent and daughter relations and their labels during the training phase where the information about the intermediate items between parents and daughters that we use is not recorded.

Table 3. Parser performance of MaltParser and NDParser

Parser	UAS(%)	LAS(%)	LS(%)	Speed
MaltParser	75.2	70.0	92.2	0.144
NDParser	74.5	71.0	93.6	0.081

The results for MaltParser in Table 3 were obtained by training and testing it on the dependency trees that we extracted from the PATB. The structure of these trees depends on the head-percolation table that is used during the conversion process. It is likely that this underlies the differences between our results for MaltParser and the results published by Nivre (2008) (77.76% for UAS, 65.79% for LAS, and 79.30% for LS), where Niver's result for UAS is slightly better than the one we obtained in this study, for LAS and LS they are slightly worse.

8 Conclusions

In this paper we have presented a simple approach for evaluating and generating parse models from various machine learning classifiers. We have shown that generating a parse model, which is used for guiding data-driven parsers, from different classifiers affects parsing performance in different ways. We have discovered that generating a parse model using a small set of features and settings improves parsing accuracy and speed compared with using large features and settings. We have presented a basic shift-reduce parser, which is based on the arc-standard algorithm of MaltParser, and we have evaluated our parser with various parse models that were generated from different classification algorithms, features and settings.

Acknowledgment

This work was funded by the Qatar National Research Fund (grant NPRP 09-046-6-001).

References

Chang, C.-c. and Lin, C.-J. (2001), Libsvm: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 3(2):1-27.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Written, I. H., (2009), The weka data mining software: An update, *SIGKDD Explorations*. *Newsl*, 11(1):10-18.

Kuhlmann, M. and Nivre, J., (2010), Transition-Based Techniques for Non-Projective Dependency Parsing, *Northern European Journal of Language Technology,* 2(1):1-19.

Nivre, J. (2008), Algorithms for deterministic incremental dependency parsing, *Computational Linguistics,* 34(4): 513-553.

Kay, M. (1973) The MIND System. In Rustin R. (Ed) Natural *Language Processing*, pp. 155–188. Algorithmics Press, New York.

Jaf, S. F. and Ramsay, A. (2013), Towards the Development of a Hybrid Parser for Natural Languages. In Jones, A. V. and Ng, N., editors, *2013 Imperial College Computing Student Workshop*, volume 35 of *Open Access Series in Informatics (OASIcs)*, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. Pp. 49–56.

Maamouri, M. and Bies, A. (2004) Developing an Arabic treebank: methods, guidelines, procedures, and tools. In Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages. Geneva, pp. 2–9.

Nivre, J., Hall, J. and Nilsson, J. (2006), MaltParser: A Data-Driven Parser-generator for Dependency Parsing, In Proceedings of LREC.

Nivre, J., Rimell, L., McDonald, R. and Gomez-Rodr´ıguez, C. (2010), Evaluation of dependency parsers on unbounded dependencies. In Proceedings of the 23rd International Conference on Computational Linguistics, COLING'10. Beijing, pp. 833–841.

Quinlan, J. R., (1992), Learning with continuous classes, In Proceedings of the 5th Australian Joint Conference on Artificial Intelligence. Sydney. pp. 343–348.

Jasnopis – A Program to Compute Readability of Texts in Polish Based on Psycholinguistic Research[1]

Łukasz Dębowski[1], Bartosz Broda[1], Bartłomiej Nitoń[1], and Edyta Charzyńska[2]

[1] Polish Academy of Sciences, Institute of Computer Science,
ul. Jana Kazimierza 5, 01-248 Warsaw, Poland
ldebowsk@ipipan.waw.pl
bartosz.broda@gmail.com
bartek.niton@gmail.com

[2] University of Silesia, Institute of Pedagogy
ul. Grażyńskiego 53, 40-126 Katowice, Poland
edyta.charzynska@us.edu.pl

Abstract. Readability of a text is a measure how difficult the text is to understand on average. The aim of the present paper is twofold. First, we have determined through a psychological experiment and statistical data analysis how readability of texts in Polish depends on syntactical and lexical statistics of the texts. Second, we have implemented a computer program, called Jasnopis, which computes readability of a given text according to the developed formula and suggests how to make the text easier to comprehend.

1 Introduction

Texts that circulate in public discourse, such as fairy tales, press articles, legal acts, or scientific articles, vary with respect to their ease of reading, called readability. Some degree of text difficulty stems from an intention of communicating complex meanings, but ideally we would expect that the intended ideas were put across as simply as possible. The reality is far from this ideal state. We are surrounded by more and more complex texts, such as legal decisions or medical leaflets, which we are supposed to understand. Unfortunately, these important texts are difficult to understand to nonspecialists, for they are written using a specific language register and contain very long sentences or highly specialized terms. We think that there is a need for a computer application that would help to measure how difficult a given text is to comprehend and would suggest possible ways of simplifying it. Consequently, in this paper we propose a computational method of predicting readability of texts in Polish.

[1] The research reported in this paper has been funded by the NCN grant "Mierzenie stopnia zrozumiałości polskich tekstów użytkowych" no. 2011/03/BHS2/05799.

A great deal of research concerning readability of texts has been already done for English. Since the end of 19th century, researchers in the United States have been interested in variation of texts with respect to their ease of understanding (Sherman, 1893). In the 1920's the interest of readability researchers shifted towards practical application such as assessing difficulty of textbooks and adjusting them to progressing abilities of schoolchildren. Hence, various empirical formulas have been proposed that allowed to estimate readability of a text given its certain statistics (Lively and Pressey, 1923; Washburne and Vogel, 1928; Lewerenz, 1929; Patty and Painter, 1931). The next decade brought interest in measuring actual understanding by adults through psychological tests and using results of the experiments to scale and improve readability formulas (Dale and Tyler, 1934; Gray and Leary, 1935). In 1940's, Lorge (1944a,b) and Flesh (1948) observed that readability can be predicted surprisingly well using only two or three statistics related to syntactical and lexical text complexity, such as the average sentence length (ASL) and the average word length (AWL). Let us observe that restricting ourselves to the AWL, we would treat the text as a bag of words. In contrast, taking into account the ASL, we indirectly measure also the complexity of syntax, which is a desirable property.

An example of a simple readability predictor is the Flesh formula (Flesh, 1948):

$$\text{Text Readability} = 206.835 - 1.015 * \text{ASL} - 84{,}6 * \text{AWL}, \tag{1}$$

where ASL – average sentence length (in words), AWL – average word length (in syllables). The readability index given above ranges between 100 (a very simple text) and 0 (a very difficult text). Many more similar readability formulas have been advocated by various researchers since then (Dale and Chall, 1948; Gunning, 1952; McLauglin, 1969; Caylor et al., 1973; Kincaid et al., 1975). The FOG index by Gunning (1952) became particularly famous. It reads:

$$\text{Fog Index} = 0.4 * (\text{ASL} + \text{PHW}), \tag{2}$$

where ASL – average sentence length (in words), PHW – percentage of words longer than two syllables. Two other popular readability indices are ARI (Senter and Smith, 1967) and LIX (Bjornsson, 1968). ARI (Automated Readability Index) was proposed for English and it reads $\text{ARI} = -21.43 + 0.5 * \text{ASL} + 4.71 * \text{AWL}$, where ASL – average sentence length (in words), and AWL – average word length (in characters). In contrast, LIX (Lasbarhetsindex) was designed for Swedish and it reads $\text{LIX} = \text{ASL} + \text{PHW}$, where ASL – average sentence length (in words) and PHW – percentage of words longer than 6 letters.

As we can see, each of the proposed readability formulas uses a bit different text statistics, with different coefficients, and returns values in a different range. Thus, there is a problem of putting readability indices onto a common human-readable scale. To overcome this problem, Dale and Chall (1948) proposed to scale readability index according to the number of years of education that is needed by the intended reader of the text. Another way of putting the readability index onto a common scale is to use some standardized and universal psychological test of text understanding and to construct the best predictor of this test based on text statistics. In fact, Taylor (1953, 1956) developed a method, called the Cloze test, which seems to measure how well human subjects understand a given text. The Cloze test consists in asking a per-

son to complete gaps in a version of the text in which every 5th word has been deleted. The Cloze score is the percentage of gaps that have been completed correctly. It has been confirmed that the Cloze score correlates well with other psycholinguistic methods of assessing text readability (Rankin 1959; Bormuth 1966). Using the Cloze test, quality of various readability formulas was computed by DuBay (2006). For example the Pearson correlation between the Cloze score and the Flesh formula (1) is 0.91, the same result was obtained for the Fog index (2) whereas the best result, correlation 0.93, was observed for the formula by Dale and Chall (1948).

It is reasonable to expect that readability formulas should be language dependent to a certain extent. The typical length of a sentence or a word clearly depends on a language. For this reason, readability formulas, particularly those suggesting the required level of reader's education, should be tuned to a particular language, such as Polish. Until recently, there was not much interest in the readability research for the Polish language. This research area started to gain more interest in the last few years, e.g., Broda et al. (2010), but the most known readability formula was proposed by Pisarek in 1960's (as reported in Pisarek, 2007):

$$\text{Text Difficulty} = \frac{1}{2}\sqrt{ASL^2 + PHW^2}, \tag{3}$$

where ASL – average sentence length (in words), PHW – percentage of words longer than three syllables. In his 2007 paper, Pisarek also published a graphical scale for computing readability, which corresponds to a bit different formula, namely

$$\text{Text Difficulty} = \frac{ASL}{3} + \frac{PHW}{3} + 1. \tag{4}$$

Pisarek has not verified his formulas in a psycholinguistic experiment on human subjects. In contrast, we will discuss the results of a larger research project in which:

1. The Cloze test and an open-ended question test was applied to 35 texts in Polish, read by a sample of 1759 persons.
2. The results of the psycholinguistic experiment were analyzed statistically to provide a new readability formula, which is better than Pisarek's formula.
3. A computer application, called Jasnopis, was written to compute this readability formula for a given text in Polish. Besides estimating readability according to our new formula, Jasnopis returns many other text statistics for a given text and prompts how to adjust the text to make it more readable.

The organization of the paper is as follows. In Section 2 we discuss the psycholinguistic experiment. Section 3 is devoted to statistical analysis of the data and the development of a new readability formula. In Section 4 we describe the Jasnopis program. Conclusions are presented in Section 5.

2 The psycholinguistic experiment

The purpose of the psycholinguistic experiment was fourfold: a) to validate Pisarek's formula, b) to find text variables that influence text readability but are different than ASL and PHW, c) to identify psychological variables that influence text comprehension (such as reader's interest in the text), and d) to use the results of the experiment to develop a new readability formula.

Before conducting the psycholinguistic experiment, we have constructed an a priori scale of seven classes of growing text difficulty, measured in the number of required years of education to understand the text correctly. (Class 1 are texts that should be understandable by students of elementary schools, whereas class 7 are those whose comprehension requires the doctorate level of education.) Subsequently, we have compiled a corpus of texts that presumably belong to the respective text difficulty classes. The texts were chosen by the project members: psycholinguists, linguists and computational linguists. Using the FOG index (2), we have next chosen 5 most typical texts for each difficulty class. In this way we have obtained a sample of 35 texts, on which we performed the psycholinguistic experiment.

In the experiment, 1759 persons have participated: 63% female and 37% male. Participants were of diversified ages (average=35.6, standard deviation=14.65, min=15, max=87), education (from elementary to higher), occupation (including manual workers, white-collars, unemployed and pensioners), coming from villages (20%), smaller towns (28.3%), medium-size cities (28.7%) and big cities (21.7%). Each participant of the study received 2 texts to read. Each text was accompanied with a set of 5 open-ended questions or the Cloze test. The experiment was performed using the traditional pen and paper approach.

Having collected the survey results, we performed statistical analysis of the data. We found out that Pisarek's formula was highly correlated with the results of the Cloze test ($r=-0.69$, $p=0.001$) and the open-ended questions ($r=0.8$ $p=0.001$). Moreover, we found out that the test results are highly correlated not only with the variables used by Pisarek (for ASL $r_{cloze}=-0.63$, $r_{questions}=-0.71$; for PHW $r_{cloze}=-0.67$, $r_{questions}=-0.83$; $p=0.001$) but also with some other text statistics. Among the top correlated variables were: the percentages of nouns, terminology, abstract nouns, foreign words, gerunds, verbs, the ratio of nouns to verbs, and the subjective probability of words (Imiołczyk, 1987). These results suggest that using these variables we may propose a readability formula that outperforms the Pisarek or Fog indices.

3 A new readability formula

Given the psycholinguistic survey described in Section 2, we were in position to analyze how readability of a text depends on particular text statistics. At our disposal we had 35 texts – 5 texts per each of 7 difficulty classes. For each text, we had the results of two psycholinguistic tests measuring the text comprehension – the Cloze test and the open question test. These were our response variables. Moreover, for each text, we have measured 33 lexical and syntactical text statistics, such as ASL,

PHW, the percentages of nouns, terminology, abstract nouns, foreign words, gerunds, verbs, the ratio of nouns to verbs, and the subjective probability of words. These were our explanatory variables. The goal of the consecutive data analysis was to find out (i) how the difficulty class of a text could be predicted from the response variables and (ii) how the response variables could be predicted from the explanatory variables. As a result, we obtained a new formula for text readability which was implemented in the Jasnopis tool, to be discussed in the next Section 4.

Since there were not so much data, we looked for a linear formula for readability:

$$Y_i = A_0 + \sum A_j X_{ij} + \varepsilon_i \quad . \tag{6}$$

where: Y_i – a chosen response variable for the i-th text, X_{ij} – j-th explanatory variable for the i-th text, ε_i – random noise, i=1,...,N, N=35 – the number of texts, K=33 – the number of explanatory variables.

The number of texts N=35 is close to the number of explanatory variables K=33. In this situation, choosing the coefficients A_j through least squares regression would lead to terrible overfitting, that is, formula (6) would not predict comprehension of texts different to the training sample. A possible solution to this problem is to use least squares regression with regularization, such as Lasso or Ridge regression (Tibshirani, 1996; Tikhonov, Arsenin, 1977). The least squares regression with regularization consists in choosing such coefficients A_j that minimize expression

$$\sum \left[Y_i - A_0 + \sum A_j X_{ij} \right]^2 + \beta \sum \left(A_j \right)^\alpha \quad , \tag{7}$$

where $\alpha = 1$ for the Lasso regression and $\alpha = 2$ for the Ridge regression, whereas β is chosen by cross validation. (For the least squares regression without regularization, we minimize expression (7) with $\beta = 0$.)

A priori it was not obvious that the Lasso or Ridge regression would give the best results. Therefore we compared these two methods with three other methods:

1. The baseline model: Text difficulty does not depend on text. That is, we minimized expression (7) with $\beta = 0$ and A_j being nonzero only for $j = 0$.
2. The least squares regression with two explanatory variables, ASL and PHW, as in Pisarek's formula (4). That is, we minimized expression (7) with $\beta = 0$ and A_j being nonzero only for $j = 0,1,2$.
3. The weighted average (committee) of least squares regressions with three explanatory variables: ASL, PHW, and one of the remaining 31 variables. That is, first, for each k in range {3,...,K}, we minimized expression (7) with $\beta = 0$ and A_j being nonzero only for $j = 0,1,2,k$, and second, we took an average over k of so obtained A_j.

The quality of each of these five methods of determining coefficients A_j was assessed by leave-one out cross validation. That is, we removed one text from the training sample, we fitted the coefficients A_j to the remaining texts, and we checked how well the model predicted the response variable for the removed text. The prediction error, defined as difference between the prediction and the response variable, was recorded for each text. We made a boxplot graph of the prediction error and we chose the method for which the prediction error is the smallest in general.

We applied this procedure independently to three response variables: the Cloze score, the open question score, and a weighted average of these two scores, which we will refer to as the weighted readability score. The relative prediction errors for predicting the Cloze score and the open question score were similar whereas they were substantially smaller for the weighted readability score. Therefore we suppose that the weighted readability score is a better predictor of the actual text readability than the Cloze score or the open question score considered individually. The boxplots of the prediction error for the weighted readability score and the five different methods of determining coefficients A_j are presented in Fig. 1.

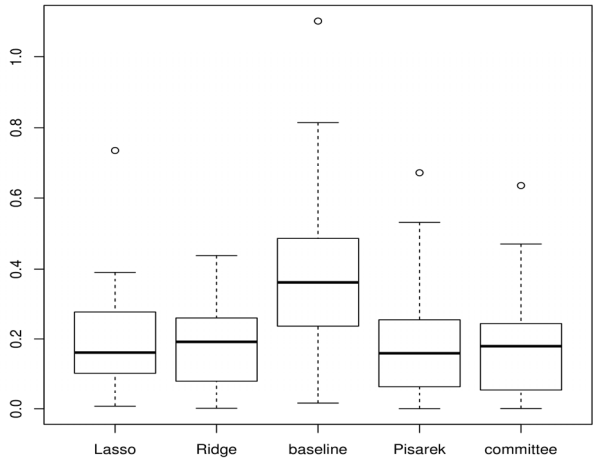


Fig 1. Boxplots of prediction error of the weighted readability score for the five methods of determining coefficients A_j described in Section 4

In Fig. 1 we can see that the Ridge regression yields the smallest maximal prediction error. Therefore formula (6) with coefficients A_j given by the Ridge regression for the weighted readability score was adopted as a part of a new formula for readability of texts implemented in the Jasnopis tool. The second ingredient of the new Jasnopis formula for readability is a projection of the Rigde regression onto the scale of 7 difficulty classes, introduced in Section 2, so that the final readability score be more human readable. As we can see in Fig. 2, the dependence between the weighted readability score and the difficulty class is linear in a good approximation.

4 The Jasnopis program

The final aim of our project was to construct a computer application for measuring readability of Polish texts. The tool, called Jasnopis, implements the measure of text readability given by the Ridge regression, described in the previous section, and addi-

tionally computes a number of other text statistics, which may be of interest to end-users. The prototype web-based application is available at http://jasnopis.pl. Jasnopis accepts many different documents types on its input: from a plain text and a website URL address to document formats supported by OpenOffice. The first step of the document processing by Jasnopis is text extraction, which is a nontrivial problem on its own. Next, we perform morphological analysis using Morfeusz (Woliński, 2006, and part-of-speech tagging using WCRFT (Radziszewski, 2013). In the later stages of document processing we use various tools and resources like the frequency lists from National Corpus of Polish, NCP (Przepiórkowski et al., 2012), the plWordnet (Piasecki et al., 2009) or the subjective probability lists of Imiołczyk (1987).

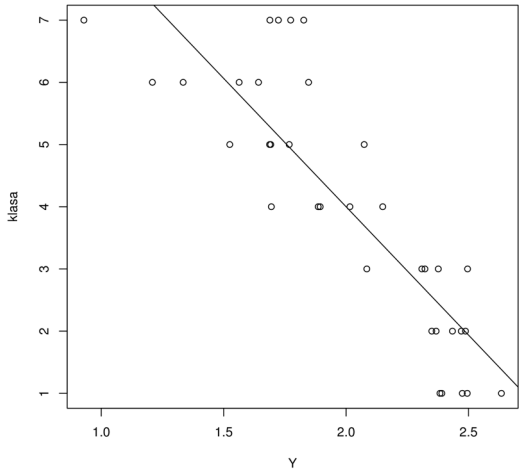


Fig. 2. The dependence between the weighted readability score (Y) and the difficulty class (klasa) for 35 texts

The main statistic calculated by Jasnopis is the difficulty class on the 7-point scale, described in Section 2. To estimate the difficulty class we rescale the score of the Ridge regression via the linear function depicted in Fig. 2. Additionally, Jasnopis calculates the following indices: a few variants of the FOG index (2), Pisarek's indices (3) and (4), an automated Taylor test, similarity graphs and additional text statistics. Both the FOG and Pisarek indices depend on the variable PHW – the percentage of words longer than a certain number of syllables (for Polish, three). Since Polish is an inflective language, it is not a priori obvious whether when computing PHW one has to consider the orthographic forms or the base forms of words.

In addition to the above variants of the FOG and Pisarek indices we also calculate some discounted versions of them, because the original definition of "hard" words in PHW is too simplistic: not every long word is a difficult word. Many words that are long are so common, that an average person has no difficulty in understanding them. Thus, we exclude most frequent words in NCP from PHW calculation. There are many words that an average Polish native speaker knows, but which are rarely used

in writing. Thus, from the PHW calculation, we also remove words that can be found on the subjective probability lists by Imiołczyk (1987).

Another text statistic implemented in Jasnopis, the automated Taylor test is somewhat inspired by the Cloze test by Taylor (1953, 1956). Instead of using human subjects, we train a few statistical language models and we check which one is the best at predicting the text. For simplicity, we use bigram language models (Jurafsky & Martin, 2008), trained on 5 reference corpora corresponding to different classes of text difficulty. Each bigram model assigns a probability of a word w_i conditioned on a single previous word w_{i-1} as

$$p(w_i|w_{i-1}) = \frac{c(w_{i-1}w_i)}{\sum_{w_i} c(w_{i-1}w_i)}, \tag{5}$$

where $c(w_{i-1}w_i)$ denotes the number of times the bigram $w_{i-1}w_i$ occurred in a training corpus. In this way we obtain seven bigram models corresponding to the seven classes of text difficulty. Then, the difficulty class of a given new text is determined as the class of difficulty corresponding to the bigram model with the highest total probability (that is, the lowest perplexity).

Yet another automated score of text difficulty implemented in Jasnopis is also based on reference corpora. Namely, instead of building language models we use the Vector Space Model (Salton et al., 1975). In this approach the text is represented as the n-dimensional vector $D=[d_1,d_2,...,d_n]$, where d_i are frequencies of words appearing in the text. To compare two texts or corpora we use the cosine distance between the corresponding vectors. Subsequently, the difficulty class of a given new text is determined as the class of difficulty corresponding to the reference corpus with the highest smallest cosine distance. Let us observe that this procedure ignores syntactic difficulty of the text since we treat the text as a bag of words. Thus, we only compare texts on a lexical level. Nonetheless, as we have determined, the lexicon is an important factor in measuring readability of a given text.

We have experimentally verified performance of both the automated Taylor test and the similarity model. Using leave-one-out cross validation we achieved from 68.31% to 100% (depending on the reference corpus difficulty class) precision for automated Taylor test and from 71.74% to 100% for similarity-based approach. See Broda et al. (2014) for details.

Besides returning a number of text statistics, Jasnopis supports also computer-aided text simplification. In a given text, it marks difficult paragraphs, too long sentences, and hard words. For hard words, substitution suggestions are presented sometimes using synonyms, hyponyms and hyperonyms from the plWordNet. No word-sense disambiguation is implemented, so the user has to make the final decision. Since simplifying a document in a web-based environment might not be very convenient, we have developed also Jasnopis plugins for OpenOffice and MS Word that cover most important functionalities of the web application.

5 Concluding remarks

In this paper we have presented an approach for constructing a new readability formula for Polish, based on Ridge regression. We use 33 lexical and syntactic text variables for predicting the text difficulty class, which is an improvement over the received readability formulas, which only use two variables. The regression coefficients in the Ridge regression were fitted to the empirical text comprehension data – a psycholinguistic experiment with 35 texts and 1759 subjects. We have also presented a computer program for measuring text readability. The application, called Jasnopis, not only implements the new formula but also provides other methods for measuring readability, both new and standard. By showing difficult sentences and words in a text, Jasnopis supports computer-aided text simplification, as well.

The proposed approach to measuring readability can be extended in several ways. One might search for additional explanatory text features. Especially, sophisticated syntactic features based on parse trees might provide additional benefits. Also, one could use other machine learning approaches to come up with even smaller error rates. Since Jasnopis already provides a few different methods for measuring readability, a straightforward approach would be to combine them using for example bagging. Last, but not least, having the ability to measure readability for Polish is a necessary step for (semi) automatic text simplification, which is an obvious direction of further research.

References

Bjornsson, C. H. (1968) *Lasbarhet.* Liber, Stockholm.

Bormuth, J. (1966) Readability: A new approach. *Reading Research Quarterly,* 1:79-132.

Broda, B., Maziarz, M., Piekot, T., Radziszewski, A. (2010) Trudność tekstów o Funduszach Europejskich w świetle miar statystycznych. *Rozprawy Komisji Językowej Wrocławskiego Towarzystwa Naukowego*, 37.

Broda, B., Ogrodniczuk, M., Nitoń, B., Gruszczynski, W. (2014) Measuring Readability of Polish Texts: Baseline Experiments. In: *Proc. of the 9th International Conference on Language Resources and Evaluation, Reykjavik, Iceland*.

Caylor, J. S., Stitch, T. G., Fox, L. C., Ford, J. P. (1973) Methodologies for determining reading requirements of military occupational specialties. Technical Report 73-5, Human Resources Research Organization, Alexander, Virginia.

Dale, E., Chall, J. S. (1948) A formula for predicting readability. *Educational Research Bulletin*, 27:1-20, 37-54.

Dale, E., Tyler, R. (1934) A study of the factors influencing the difficulty of reading materials for adults of limited reading ability. *Library Quarterly*, 4;384-412.

DuBay, W. H. (2006) *Smart Language: Readers, Readability, and the Grading of Text.* Impact Information, Costa Mesa.

Flesch, R. (1948) A new readability yardstick. *Journal of Applied Psychology*, 32:221-233.

Gray, W., Leary, B. (1935) *What Makes a Book Readable.* University of Chicago Press, Chicago.

Gunning, R. (1952) *The Technique of Clear Writing.* McGraw-Hill, New York.

Imiołczyk, J. (1987), *Prawdopodobieństwo subiektywne wyrazów: podstawowy słownik frekwencyjny języka polskiego.* Warszawa.

Jurafsky, D. and Martin J. H. (2008). Speech and Language Processing (2nd Edition). Pearson Prentice Hall.

Kincaid, J. P., Fishburne, R. P., Rogers, R. L., Chissom, B. S. (1975) Derivation of new readability formulas (Automated Readability Index, Fog Count, and Flesch Reading Ease Formula) for Navy enlisted personnel. CNTECHTRA Research Branch Report, pp. 8-75.

Lewerenz, A. S. (1929) Measurement of the diffi-culty of reading materials. *Los Angeles Educational Research Bulletin*, 8:11-16.

Lively, B. A., Pressey, S. L. (1923) A method for measuring the 'vocabulary burden' of textbooks. *Educational Administration and Supervision*, 9:389-398.

Lorge, I. (1944a) Predicting readability. *Teachers College Record*, 45:543-552.

Lorge, I. (1944b) Word lists as background for communication, *Teachers College Record*, 45:543-552.

McLaughlin, G. H. (1969) SMOG grading—a new readability formula, *Journal of Reading*, 22:639-646.

Patty, W. W., Painter, W. I. (1931) A technique for measuring the vocabulary burden of textbooks , *Journal of Educational Research*, 24:127-134.

Piasecki, M., Szpakowicz, S., Broda. B. (2009) A wordnet from the ground up. Oficyna wydawnicza Politechniki Wroclawskiej.

Pisarek, W. (2007) O mediach i języku. In: *Jak mierzyć zrozumiałość tekstu?*, pp.245-262. Universitas, Kraków.

Przepiórkowski, A., Bańko, M., Górski, R. L., and Lewandowska-Tomaszczyk, B. (2012). Narodowy Korpus Języka Polskiego. Wydawnictwo Naukowe PWN, Warsaw

Radziszewski, A (2013). A tiered CRF tagger for Polish. In: *Intelligent Tools for Building a Scientific Information Platform: Advanced Architectures and Solutions*. Springer Verlag.

Rankin, E. F. (1959) The cloze procedure – Its validity and utility. In: Causey O., Eller W. (eds.) *Eighth Yearbook of the National Reading Conference*, pp. 131–142.

Sherman L. (1893) *Analytics of literature: A manual for the objective study of English prose and poetry*. Ginn and Co, Boston.

Taylor, W. L. (1953) Cloze procedure: a new tool for measuring readability. *Journalism Quarterly*, 30:415–433.

Taylor, W. L. (1956) Recent developments in the use of 'cloze procedure'. *Journalism Quarterly*, 33:42–48.

Tibshirani, R. (1996) Regression shrinkage and selection via the lasso, Journal of the Royal Statistical Society, series B, 58:267-288.

Tikhonov, A. N., Arsenin, V. Y. (1977) *Solution of Ill-posed Problems*. Winston & Sons, Washington.

Salton, G., Wong, A. and C. S. Yang (1975) A vector space model for automatic indexing. Communications of ACM, 18(11):613–620.

Senter, R. J., Smith, E. A., (1967) Automated Readability Index. Wright-Patterson Air Force Base, p. iii. AMRL-TR-6620.

Washburne, C., Vogel, M. (1928) An objective method of determining grade placement of children's reading material, *Elementary School Journal*, 28:373-381.

Woliński M., 2006. Morfeusz – a Practical Tool for the Morphological Analysis of Polish. Intelligent Information Processing and Web Mining, Springer.

A Logical Form Parser for Correction and Consistency Checking of LF resources

Rodolfo Delmonte, Agata Rotondi

Department of Linguistic Studies and Comparative Cultures and Department of Computer Science
Ca Foscari University - 30123 VENEZIA (Italy)
delmont@unive.it

Abstract. In this paper we present ongoing work for the correction of Extended WordNet (XWN), the most extended freely downloadable resource of Logical Forms (LFs) – by the Human Language Technology Research Institute (HLTRI) of University of Texas at Dallas (UTD). In a previous paper we reported on type and number of errors detected in the 140,000 entries of the resource, which amounted to some 30%. This didn't include problems related to inconsistencies from disconnected variables which were not computable at the time. We now created an LF parser that parses each entry after appropriate transformations. The parser has been created to count the number of disconnected variables, be they object variables or predicate event variables: the result is 56% of LFs containing some disconnected variable. We devised two procedures for correction: one lexical and the other structural which eventually allowed a dramatic reduction: the final count is now 24%. Additional work has been carried out to improve the general consistency by manual intervention on "inconsistent" outputs signaled by the parser and has reduce the number of errors to a reasonable percentage for such a resource, that is less that 15%.

1 Introduction

This paper presents work carried out to parse and correct LF resources and in particular XWN or Extended WordNet (see Mihalcea et al. 2001), a freely downloadable resource containing a mapping into Logical Forms (LFs) of WordNet glosses. We started to produce a parser of LFs after working at individuating errors in XWN (see XXX). After we discovered that some 30% of all entries needed some corrections, we decided to continue work for a number of reasons, some of which positive some others negative that will be discussed below.

As to negative question, the first regards the way in which syntactic structure has been produced. Current parsers mostly produce surface dependency/constituency structure with good enough approximation, but which is of no use for the mapping into LFs seeing that deep relations are missing. We are here referring to two types of parsers, Charniak's constituency-like parsers and dependency-like parsers. Deep parsers are only a few and the accuracy of their performance is insufficient for a mapping into LFs seeing that LFs require fully consistent representations in order to preserve the semantics (but see below).

Now, there is a number of applications that produce LFs directly from syntactic representations, but their performance is unsatisfactory for the reasons explained in the previous paragraph. We tried LFToolkit (Rathod, Hobbs, 2005) which maps into LF directly from the output of Charniak's parser - more on this below. Worthwhile mentioning is also Cahill et al. (2007) attempt at transforming a portion of WSJ into LFs by means of a conversion of PennTreebank II augmented syntactic representation into complete F-structures. The authors claim an F-measure of 97% over the 96% of sentences converted, which is certainly a success. However, WSJ sentences are in no way comparable to glosses and their online parser does not allow the creation of LF[1]. Since LFs cannot be produced fully automatically and need a lot of manual additional work, we thought it reasonable to try and use existing LF resources such as ILF (Intermediate Logical Forms) - see Agerri & Peñas, 2010, XWN and others. We believe it is always worthwhile correcting these resources, wherever possible. WordNet glosses are definitions, meaning paraphrases and declarative descriptions associated to synsets of WN, which is what makes them highly valuable for semantically heavy tasks such as Q/A, WSD, and Text Understanding in general.

Coming now to the actual resources, LF mapping from PT (Penn Treebank)-like constituency-based syntactic structures are - in our opinion - a lot more error prone than those derived from dependency structure (see Agirre & Peñas, 2010). This is due to the fact that PT-like structures are more difficult to map due to the nature of constituency structure, which is more functionally based than semantically oriented, when compared to dependency structures. Syntactic constituency in PT as well as the one produced by Charniak's parser, associates main constituency nodes to functional heads like auxiliaries, complementizers, subordinating conjunctions, relative and interrogative pronouns, modals, verb particles. Nominal heads are usually lumped together with their determiners and modifiers, be they other nouns or adjectives. For this reason, using a dependency structure in which semantic heads are separated from functional ones can be and is – as ILF clearly shows, but see below – highly beneficial for a safer mapping into LF. In this sense, ILF being mapped from dependency structures is much closer to semantic content than XWN - more on ILF below.

Both XWN and ILF have been mapped without the help of additional resources such as lexica and anaphora resolution algorithm, which were in fact necessary, as will be shown below. However, the net result is the absence of free ungrounded variables in ILF: on the contrary in XWN, the presence of ungrounded variables is the rule, as will be shown below. This is partly due to the lack of any one to one correspondence between constituency structure and LF as encoded in the mapping algorithm.

The paper is organized as follows: in section 2 we will review different types of Logical Forms and try to evaluate the contribution of each representation; in section 3 we concentrate on XWN, WNE and ILF and individuate their strengths and weaknesses; section four is dedicated to presenting the parser itself; then we end with some evaluation, conclusion and future work.

[1] http://lfg-demo.computing.dcu.ie/lfgparser.html

2 Logical Forms, but what kind?

In XWN, WNE and ILF the mapping to LFs has been done semi-automatically with manual checking of the majority of syntactic constituency structures. However, the actual mapping process has not been subsequently checked nor evaluated (but see Vasile, 2004).

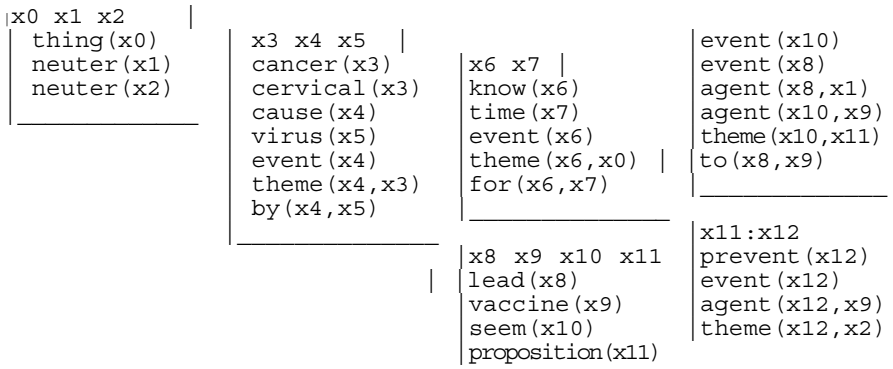
What kind of LF are we referring to? The LF we are referring to is a flat unscoped first-order logic (FOL) well-formed formula representing the "meaning" of a sentence. It has been restricted to a conjunction of predicates which in turn contain arguments that have been hampered from being themselves recursive. Possible arguments of predicates can be event variables and argument variables, the latter being also called object variables, referring to entities, properties and attributes.

However, not all work on Logical Form would look the same and there are lots of different ways of computing and building them. In XWN, for instance, there is no attempt at covering all if not most of what is commonly regarded as semantically related problems that might as such be represented in a LF. Here below we show some valuable attempt at including some of the semantics in the LF from the contents of the book by (Bos & Delmonte, 2008) for the workshop of ACL Sigsem held in Venice.

The first LF we show is the one used by J.Bos to represent DRS. As can be seen below, variables introduced in the representation are all of the same kind, the prefix always being X. What changes is the number that follows the X. As a result there is no distinct event variable, with an E prefix. The text is one of the Shared Task of the workshop and we take these two sentences (ibid., 283):

Sent.1 Cervical cancer is caused by a virus.

Sent.2 That has been known for some time and it has led to a vaccine that seems to prevent it.



In this LF, events have been reified and appear as functions heading their variable. Also semantic/thematic roles have been reified, and head the variables of both argument and event. This choice multiplies the number of logical objects, but simplifies the matching. Another way of rendering the LF of sentence 1, could have been

```
cause(e4,x5,x3),cancer(x3),cervical(x3),by(e4,x5),virus(x5)
```

A slightly similar approach has been taken by (Clark et al., 2008) with a system that also comprises a parser and a logical form generator[2]. Their example is shown below, where variable are indicated by underscored _X:

```
Sent.3 "A soldier was killed in a gun battle."
(DECL ((VAR _X1 "a" "soldier")
(VAR _X2 "a" "battle" (NN "gun" "battle")))
(S (PAST) NIL "kill" _X1 (PP "in" _X2)))
```

This mixed syntactic structure is then used to generate "ground logical assertions of the form r(x,y), containing Skolem instances, by recursively applying a set of syntactic rewrite rules to it. Verbs are reified as individuals, Davidsonian-style."(ibid., 48; but see also Davidson, 1967;1980):

```
object(kill01,soldier01)
in(kill01,battle01)
modifier(battle01,gun01)
```

As the authors comment, predicates used in this representation are just syntactic relations of the type SUBJect_of, OBJect_of, and MODifier_of and all prepositions, which typically take two variables related to the individuals they are bound to. In particular, in this representation Skolem instances are associated with its corresponding input word. Syntactic relations represent deep relations: the surface subject of the passive sentence Sent.3 is turned into an OBJect.

Another richer way of representing meaning in LF is proposed by Delmonte in Bos & Delmonte 2008:291, for the sentence,

```
Sent.4 John went into a restaurant
wff(situation,
    wff(go,
     < entity : sn4 : wff(isa, sn4, John) >,
     < indefinite : sn5 : wff(isa, sn5, restaurant) >,
     < event : f1 : wff(and, wff(isa, f1, ev),
       wff(time, f1, < definite : t2 :
    wff(and, wff(isa, t2, tloc), wff(pres, t2)) >)) >))
```

where we see that two semantic elements appear in the representation, DEFINITENESS, and TENSE which is associated to a Reference Time location vari-

[2] In the authors' words, the LF is "a semi-formal structure between a parse and full logic, loosely based on Rathod & Hobbs, 2005. The LF is a simplified and normalized tree structure with logic-type elements, generated by rules parallel to the grammar rules, that contains variables for noun phrases and additional expressions for other sentence constituents. Some disambiguation decisions are performed at this stage (e.g., structural, part of speech), while others are deferred (e.g., word senses, semantic roles), and there is no explicit quantifier scoping. "

able, T2. As will be discussed below, Reference Time and Definiteness are two important semantic features and are introduced also in other LF representations.

3 Previous Work related to XWN

There is a certain amount of additional work carried out on XWN that we want to review briefly here below. Apart from XWN by UTD people, the other effort to translate WN Glosses into Logical Form is by people at USC/ISI California, in 2006. Their results are also available on the web and freely downloadable at ISI, 2007. As the comment on the related webpage clearly states, "The following additional "standoff" files providing further semantic information to supplement the WordNet 3.0 release." The file contains LFs in XML format for most of the glosses "except where generation failed" as the comment clearly warns out. The authors made a subset of the core WordNet including 2800 noun senses in plain text format, in 2007. As the comments on the website say, "these are generally of higher quality than those contained in the file below for all glosses." We find these representations in eventuality notation too cluttered with additional event variables, which makes the LF entry too heavy to read, as can be seen in the example of the entry BUTTER included below. These files can be downloaded at http1;http2. The conversion process of WN glosses proceeds by parsing with Charniak parser and the result is converted into a logical syntax by a system called LFToolkit (Rathod & Hobbs, 2005). Each lexical semantic head is transformed into logical fragments involving variables. For instance "John works" - commented in detail below - is translated into John(x1) & work(e,x2) & present(e), where e is a working event. Object variables are differentiated at first (x1 and x2), and then a rule which recognizes “John” as the subject of “works” sets x1 and x2 equal to each other. This works for the majority of English syntactic constructions. As the authors themselves comment, whenever there was a failure by the LFToolkit, it happened as a result of a bad parse, due to the presence of constructions for which no rule in LFToolkit had been written[3].

I will show here below the entry for BUTTER as it has been transformed by the two systems. The first representation is the one produced at USC/ISI and the second one is the one by XWN in (Moldovan & Rus, 2001). In fact both representations are in XML format, but for easiness of reading we eliminate angled brackets:

[3] "In these cases, the constituents are translated into logic, so that no information is lost; what is lost is the equalities between variables that provides the connections between the constituents. For instance, in the “John works” example, we would know that there was someone named John and that somebody works, but we would not know that they were the same person. Altogether 98.1% of the 5,000 core glosses were translated into correct axioms (59.4%) or axioms that had all the propositional content but were disconnected in this way (38.7%). The remaining 1.9% of these glosses had bizarrely wrong parses due to noun-adjective ambiguities or to complex conjunction ambiguities."(ibid.,49)

```
example (1)
entry word="butter#n#1" status="partial"gloss = an
edible emulsion of fat globules made by churning milk
or cream; for cooking and table use butter#n#1'(e0,x0)
-> edible'(e9,x1) + emulsion#n#1'(e1,x1) + of'
(e6,x1,x12) + fat#n#1'(e15,x17)+nn'(e14, x17,x12) +
globule#n#1'(e10,x12) + dset(s5,x12,e10+e14) + ma-
ke#v#15'(e2,x4,x3,x2) + by'(e3,x5,e7) + churn#v#1'
(e7,x10,x14) + milk/cream#n#2'(e11,x14) + for'(e4,x6,
x11) + cooking'(e12,x16) + table'(e13, x15)milk'
(e11, x14) -> milk/cream#n#2'(e,x14)cream#n#2'(e11,
x14) -> milk/cream#n#2'(e,x14)

example (2)
butter:NN(x1) --> edible_JJ(x1) emulsion:NN(x1)of:IN
(x1,x2) fat:JJ(x2) globule:NN(x2) make:VB(e1,x9,x1)
by:IN(e1,e2) churn:VB(e2,x2,x5) milk:NN(x3) or:CC
(x5,x3,x4) cream:NN(x4)
```

As can be seen in example(1), all lexical items are treated as predicates and have an event variable starting with E, associated to them. Event variables are typically unbound and should be quantified over. They are associated to object variables which start with X. In some cases, when a DSET is asserted, event variables are connected explicitly to their object variable, as in the nominal compound "FAT GLOBULES". Also this LF representation, which is classified as PARTIAL, contains a lot of unbound or ungrounded variables, as for instance in the case of MAKE(e2,x4,x3,x2) in example (1), where none of the object variables have an individual ground object linked to them.

Example(2) is much simpler and shorter. In this case, the LF representation produced has a better output. However, if we look at the representation associated to MAKE, we notice that only has three variables, one of which is the event variable, e1, and the remaining two are argument variables - x9, x1. Whereas x1 is properly bound to the entry BUTTER, the second variable is unbound. We can also notice that the first representation treats MAKE as a 3-place predicate, as in the sentence "John made the butter smooth by...". On the contrary, the second representation only has two argument variables: this could be justified by the use of MAKE in a participial structure, with a different meaning though. The meaning in this case is obtained by omitting the secondary predication. It is just a simple transitive structure in a passivized form. More on this topic below.

The problem related to these examples are typical problems of mapping from surface syntactic structures to Logical Forms, and we have tried to overcome them by building an LF parser that checks for consistency. Here the term consistency is referred solely to the existence of free unbound or ungrounded variables in a given LF representation. This fact does not allow relations indicated by predicates to be associated to arguments and modifiers, which are thus disconnected. In this way, the formula is useless and meaningless. Variables associated to predicates needs to be equated with those of the arguments of the predicate in order to acquire semantic consistency. From our analysis, 54.05% of all LFs contained in XWN suffer from that

problem. In particular, they constitute 71,658 over 132,587 where we found the following data:

Table 1. Errors detected by the parser

categories	Dis.Vars	Tot.LFs	%
Adverbs	487	3982	12.23
Adjectives	8886	20317	43.74
Verbs	9751	14454	67.46
Nouns	52672	94028	56.00
Total	71,658	132,587	54.05

Here percent values refer to errors found by the parser.

4 The LF Parser

The parser takes as input two files, one containing the list of logical forms as they have been listed in XWN for the different categories - verb, noun, adjective, adverb; and another file containing the synset offset followed by the synset. Each synset is associated to a gloss that contains one or more definitions, comments or examples: this is what has been transformed into Logical Forms in XWN. So the parser knows that there may be one of more LFs to associated to the same synset offset index. Now, each Logical Form will necessarily start with the same lemma which corresponds to the first lemma in the synset: for instance, the entry 00002931 of the ADJ dataset corresponding to the synset "abducent, abducting", has the associated gloss "especially of muscles; drawing away from the midline of the body or from and adjacent part". This gloss is transformed into two LFs, respectively,

```
abducent:JJ(x1)-> draw_away:VB(e1,x1,x5) from:IN(e1,x2)
midline:NN(x2) of:IN(x2,x3) body:NN(x3) from:IN(e1,x4)
adjacent:JJ(x4) part:NN(x4)

abducent:JJ(x1) -> especially:RB(x1) of:IN(x1,x2)
muscle:NN(x2)
```

These have then been turned into the Prolog compliant corresponding structures below:

```
lf(abducent_JJ(x1),[draw_away_VB(e1,x1,x5),from_IN(e1,x2),mid
line_NN(x2),
of_IN(x2,x3),body_NN(x3),from_IN(e1,x4),adjacent_JJ(x4),part_
NN(x4)]).

lf(abducent_JJ(x1),[especially_RB(x1),of_IN(x1,x2),muscle_NN
(x2)]).
```

The parser takes the synset offset associated to the current synset and the first LF in the current list. Then it matches the first lemma in the synset with the lemma head-

ing the LF. After correcting the LF, the parser checks the rest of the list to see whether there is another occurence of the current lemma and in that case it keeps the same offset index, otherwise it passes the rest of the synset-offset list. As will be discussed in detail in a section below, this version of the algorithm works fine only for a part of WordNet, proper names behave differently and the algorithm had to be modified to cope with them. The output of the algorithm is a conjunction of the information contained in the two files, as follows:

```
synset(300002931,abducent_JJ(x1),[abducent,abducting])-
[draw_away_VB-[e1,x1], from_IN-[e1,x2],midline_NN(x2),of_IN
(x2,x3),body_NN(x3),from_IN-[e1,x4],adjacent_JJ(x4),
part_NN(x4)]

synset(300002931,abducent_JJ(x1),[abducent,abducting])-
[especially_RB(x1), of_IN(x1,x2),muscle_NN(x2)]
```

There are at least three different ways of conceiving the relation intervening between variables in a flat unscoped LF. As discussed above, the simplest way would be that of considering all variables free and each one different from the others, and then at the end, specifying those variables that have to be regarded equal by additional equations. A second way, is to regard all variables equal, and then specifying the ones that have to be regarded different - and this is what has been done in Robust Minimal Recursive Semantics, (RMRS) (Copestake, 2009). In both these two ways, however, variables need to be precisely bound as required, which is not what actually happens. We report one of the examples from that presentation, for the sentence "Some big angry dogs bark loudly", where we see that a scoped LF is used to convey the role of quantifier "some":

```
example (3)
some_q(x4, big_a_1(e8,x4) ∧ angry_a_1(e9, x4) ∧ dog_n_1(x4),
bark_v_1(e2,x4) ∧ loud_a_1(e10,e2))
```

Notice that in this LF representation, every attribute and modifier has a separate event variable name. This is remarkably different from what has been done in XWN.

The third way, which is more consistent with what has been done in the XWN LF representation, is to consider variable equalities to indicate relations of some kind: in particular, any modification relation is indicated by variable equality; the same applies to argument relations. Another important topic regards the way in which optional or omitted arguments should be treated in the LF representation. As discussed in (Copestake, 2009) LFs for predicates should consider deep structure information rather than simply surface structure, and in case an argument is missing - because it has been omitted in a passive structure or simply because optional - this should be signaled appropriately by marking the corresponding slot with U (for unexpressed). These are some of the problems that we will try to tackle in our parser.

4.1 The architecture of the LF parser

The LF parser is organized in a pipeline:

A. the first module tries to match variables in predicates with their object counterpart.

B. the second module does the opposite: it tries to match variables in object formulas with their predicate counterparts.

This has to account for a number of different logical structures. The most common one is the one accounting for predicate argument structures governed by verbs, as in,

`- buy_VB(e1,x2,x1)`

where e1 is a generic event variable which might or might not have higher level binders, or meta level formula (see below) associated to it; x2 is by slot convention a variable associated to the complement (in this case an object); and x1 is again by slot convention the variable associated to the subject, treated as external argument. A second structure is the one associated to prepositions and other similar two place relation markers as comparative conjunctions or even subordinators and relative pronouns:

`- of_IN(x2,x3) - than_IN(x4,x5) etc.`

where x(number) variables bind objects, and prepositions - when they introduce gerundives - and (subordinating) conjunctions treated as relation markers:

`- by_IN(e1,e2) - since_IN(e4,e5)`

All relation markers only contain relational variables and no event or object variable of their own.

Object formulae include simple one place predication with just one variable associated to an entity, a property or an attribute, as in

`- dog_NN(x2)`

XWN uses the same specification also for modifiers like adjectives and adverbials:

`- angry_JJ(x2), fast_RB(e2)`

but this, on the basis of what we have commented above, needs some reorganization. Modifiers are then supplemented by an additional variable of their own that accounts for the role of predicate they fulfill. This will allow to differentiate cases in which the same adjectival word - say "red" - may play the role of predicate in a copulative construction which has to be differentiated from the role of attribute in a nominal compound, as in (4b) "The red hat was stiff" vs. (4a)"The stiff hat was red". The two sentences could be differentiated as follows, where x1 is associated to the subject of predication, "hat" and the predication itself is constituted by a different property identified by variable e3. The attribute is associated to the nominal head object variable x1 and is specified with event variable e2, assuming in this way that the property of being “stiff” is independent of the property of being “red”, but they are both associated to the entity X1:

```
example(4a)
be_VB(e1,x1), hat_NN(x1), stiff_JJ(e2,x1), red_JJ
(e3,e1)
example(4b)
be_VB(e1,x1), hat_NN(x1), red_JJ(e2,x1), stiff_JJ
(e3,e1)
```

There are then mixed formulae which include both event and object variables, as in,

```
- by_means_of_IN(e1,x5) = (buying) by means of
 - consider_VB(e3,e1,x2,x1), silly_JJ(e1,x2) = consider
 (your dog) silly
```

where x2 is the variable associated to "dog" and e1 is included in the argument list of the verb. This is what differentiated real copulative verbs like "be", and transitive verbs like CONSIDER which have secondary predication as argument. As anticipated above, there are also meta-level formulae and they are of two types:

- coordinating conjunctions
- complex nominal compound

The first refers to coordinating conjunctions, which allow to refer to sets of objects or predicates. The latter are separately specified: here, rather than duplicating the noun governors, the meta abstract coordinating conjunction is used, as shown below. Coordination may interest both event and object variables, as follows:

```
- and_CC(x6,x1,x2,x3)
- decide_VB(e4,e5,x6), leave_VB(e5,x6)
```

for the sentence, "Frank, John, and the dog decided to leave".

A similar formula would be used in case event variable should be coordinated as in,

```
- or_CC(e8,e1,e2,e3)
```

where e8 would be the variable associated to the coordinate structure.

The other meta level formula is associated to the function NN introduced in the XWN following Jerry Hobb's suggestions. In this case, the only possible set of variables is the one associated to objects or nominals indicating properties of the head, usually the last variable in the set. As in one below for "Samsung's profits" where x4 and x6 are bound to single entries,

```
- NN(x7,x4,x6)
- Samsung_NN(x4), profit_NN(x6)
```

and the variable x7 would be used by the event predicate that governs the nominal compound,

```
- rise_VB(e2,x7)
```

In order to allow for smooth matching procedures, all meta-level formulae are turned into their simple binary level by reification. Reification is also used for negation as shown below:

- coordinations are turned into one place predicates,

```
- and_CC(x6,x1,x2,x3) --> and_CC(xc), coord(xc,x6),
coord(xc,x1), coord(xc,x2), coord(xc,x3)
```

- negations are reified:

```
- not_RB(e2) --> neg(xn,e2),not(xn)
```

In fact, all negation formulae had to be corrected before starting to check for their consistency. We eliminated all DO auxiliaries and associated the negation predicate directly to the main verb variable, using regular expressions. In this way we got a double result: unwanted auxiliary information was eliminated and the negation operator is now correctly associated to the main verb meaning. A similar change had to be introduced for all cases of wrong treatment of the amalgam CANNOT, which in XWN is introduced directly without a decomposition, and in many cases is wrongly tagged as noun. So we produced the following change again by regular expression, where we deleted the variable associated to the wrong tagging and substitute it with the right one.

4.2 Correcting Logical Forms

We envisage two types of corrections: one induced by lexical information and another by structural information. The first correction is addressed to all those predicates that contain a dummy variable for an argument which does not exist in reality. In fact, here we are referring to verbs belonging to classes like unergative verbs, unaccusative verbs, weather verb, impersonal verbs, but also to verbs which can be intransitivized, ergativized. That is verbs which induce intransitive structures, either by raising the object to subject position and eliminating the deep subject; or cases of verbs which allow the object to be left unexpressed, that is something which can be quantified over by an existential quantifier. Always on the basis of lexical information, we check for intransitivized and passivized TRANSITIVE verbs, which constitute by far the majority of cases. In particular, in case a passivized past participle is being used, this is usually accompanied by the omission of the deep subject.

As to the structural corrections, we have been filtering wrong structures by a procedures that allows the correction module to select only those parts of the formula which need to be modified. In order to extract information related to wrong and inconsistent LFs, the parser collects variables related to object formula separately from those related to predicate formula. Then it does a simple intersection. The set of intersecting variables is then used to verify whether there are ungrounded variables.

We used the two procedures in a sequence – at first we found ungrounded variables and then looked for predicates with unneeded variables that coincided with the ones found in the previous procedures – and eliminated them. The results are remarkable: we managed to eliminate some 32%, that is almost half of previous 72% of all disconnected variables. In particular, they now constitute 31,583 over 132,587 – 23.82%.

Table 2. Errors after parser correction

categories	Dis.Vars	Tot.LFs	% after	% before
Adverbs	250	3982	6.28	12.23
Adjectives	1599	20317	7.87	43.74
Verbs	823	14454	5.69	67.46
Nouns	28911	94028	30.75	56.00
Total	31,583	132,587	23.82	54.05

As said above, XWN introduces a first event variable e1 or sometimes e0, which should be quantified over and are left unbound. Also a first object related variable is always associated to nouns and adjectives, and it is X1. These are not considered in the intersection and are removed from the set. Here below some examples of inconsistent formula for ABLE:

gloss: having the necessary means or skill or know-how or authority to do something

```
able:JJ(x1) -> have:VB(e1, x1, x8) necessary:JJ(x8)
means:NN(x2) skill:NN(x3) know-how:NN(x4) or:CC(x8, x2,
x3, x4, x5) authority:NN(x5) to:IN(x8, e2) do:VB(e2,
x8, x6) something:NN(x6)
```

where DO has x8 as Subject variable, which should be x1, that is the person that is ABLE, SUBJect of the predication of HAVE and also head of the adjective modifier. More errors are contained in the formula, where necessary(x8) should be necessary(x2), seeing that it only modifies "means". The following case is an inconsistency caused by various errors: "dependent_on" is associated to an unground variable "x4" and not "x1"; the same applies to "relative" which is associated to "x2", rather than "x1":

gloss: not dependent on or conditioned by or relative to anything else

```
independent:JJ(x1) -> not:RB(e2) dependent_on:JJ(x4)
condition:VB(e1, x5, x1) by:IN(e1, x5) or:CC(e2, e1)
relative:JJ(x2) to:IN(e2, x2) anything:NN(x2)
else:JJ(x2)
```

here below, we show what the correct LF would be like after introducing predicate variables in each adjective modifier, changed ungrounded variable associated to obligatory argument into an undefined U variable:

```
independent_JJ(x1) -> not_RB(e3),
dependent_on_JJ(e1,u,x1), or_CC(e3,e1,e2),
condition_VB(e2,x2,x1), by_IN(x2,u), or_CC(e4,e3,e5),
relative_to_JJ(e5,x3,x1), anything_NN(x3), else_JJ(x3)
```

where DEPENDENT_ON has become a complex phrasal predicat. In its formula, the preposition ON requires an additional variable, ungrounded, though; and RELATIVE has also become a phrasal adjective with preposition and as such in need of an additional argument variable. To this end, we turned both adjectives into two place predicates with an event variable to indicate that there is a dummy BE verb implicit in the gloss.

As said above, the parser at first measures intersection: in case no intersection intervenes then a flag is written on the output file and used by the correction module. In the following case, for instance, after deleting x1 and e1, the intersection is empty.

```
INPUT
lf(approved_JJ(x1),[generally_RB(e1),especially_RB(e1),
officially_RB(e1),judge_VB(e1,x5,x1),acceptable_JJ(x3),
satisfactory_JJ(x3)]).
OUTPUT
approved_JJ(x1)  6  [x5,x3] no intersection
lf(approved_JJ(x1),[generally_RB(e2,e1),especially_RB
(e3,e1),officially_RB(e4,e1),judge_VB(e1,e5,u,x1),
acceptable_JJ(e6,e5),satisfactory_JJ(e7,e5)]).
```

where we see that APPROVED has an LF formula made of 6 elements, that the intersection of the relevant variables is empty, and that there are two variables in particular which have no correspondence in object formula. The parser will inspect the formulas one by one, and eventually will equate x5 with x3, thus making the whole LF consistent. The equation decision is determined by the fact that: x5 is contained in a predicate formula, which also contains x1, and that x3 are both contained in object formula. In addition they both FOLLOW the predicate, this being a clear indication - in English at least - that they constitute a COMPLEMENT to the predicate itself. The modified and corrected formula contains also additional event variables for attributes predicated to x3 and an U for unexpressed argument variables.

In particular, the system found 110 cases of inconsistencies in the ADVERBS file of XWN, 1154 cases of inconsistencies in the ADJECTIVES file, 2054 in the VERBS file and 5276 in the NOUNS file. Overall, 8594 cases that we addressed by the correction module. The parser managed to correct half of them in a first run. Then more rules have been devised to correct the rest of the errors. These rules have then been used to check and correct most of the remaining LFs.

In fact, as a whole, we managed to correct many more entries, thanks to the fact that the parser simply got stucked whenever the LF entry was not computable, i.e. none of the variables matched either x1 or e1. We also corrected all negation operators and some of the conjunctions which were not tagged consistently, as for instance THEN, which was tagged as RB (adverbial) most of the time, and only sometimes as IN.

The evaluation of the corrections produced manually and automatically is made directly by the parser itself. The output of the parser, in the correction mode, is a file containing all LFs which have some inconsistency. Corrections have been carried out specifically on the output of the parser. Evaluation in this case is computed accordingly on the basis of number of mistakes found by the same parser.

4.3 The case of Proper Names in WN

The parser works smoothly with three files, the ones containing Adverbs, Adjectives, and Verbs, but when the file containing Nouns is started problems arise which eventually obliged us to modify the algorithm. WordNet contains some 24K capital letter initial lemmata which can be computed as proper names or named entities, i.e. person names, organization names, famous events names, institution names, location names, etc.(see Miller & Hristea, 2006). Contrary to expectations, the description of these entries in the database resembles the one used for common nouns, which as we know are used to denote classes of individuals, whereas proper names would rather be used to individuate uniquely a single referent in the world - they are rigid designators according to Kripke[4]. In fact, this is only partially true, seeing that a proper name made by the compound of first name and surname today can be regarded ambiguous and can refer to different referents in the world. The problem was partially amended by Miller & Hristea, where they produced a new version of WordNet, 2.1 in which instances – proper names – where differentiated from classes – common nouns – by the presence of a suffix in the definition of hypernyms, added to @, like this @i (ibid. 3).

Now, let's consider synsets: synsets are a collection or set of synonym lemmata which may constitute a single concept in a specific language. Lexica of different languages may vary a lot on this and a synset made of a plurality of referent words for the same concepts, translated in another language may turn up to be uniquely denoted by one single lemma. The other dimension of synsets is that they can be used to register the presence of homographs denoting different concepts, i.e. their polysemous nature. In this case, the same lemma - in case the synset is a singleton - or the first lemma of the set – if the synset has more than one member, is used. This is marked in WN by a different synset offset index as for instance in the typical case of PLANT, which is associated to the following four synsets:

```
00014510   plant, flora, plant_life
03806817   plant, works, industrial_plant
05562308   plant
09760967   plant
```

The same word – a polysemous homograph – appears either as singleton or as first member of a synset to denote different concepts. From a lexicographic point of view these four entries, which instantiate totally different senses and are associated with different glosses, are located in different places or lexical fields. As can be seen, the offset indices are very far from one another, thus indicating the distance in meaning involved in each of the different lemma forms. This is what we find with common nouns: it would be impossible to have a duplicate of the same lemma in adjacency within the same semantic lexical field indicating an instance or a slightly different meaning. Polysemous words in WordNet are not many, and their presence in first position in the synset is also an indication of the high frequency of usage of the word in the language.

[4] In his lecture in 1970, and then published in the book Naming and Necessity, by Saul A. Kripke, 1980, Blackwell, Harvard University Press.

The problem is that WordNet uses a similar technique to store information about "polysemous" proper names. In fact, this may sound quite strange, seeing that the only meaning associated to a proper name is the referent which they should designate. So what WordNet is actually highlighting by associating a synset to proper names is, perhaps, the possibility that two or more proper names share part of the name. This is usually the last name for person names and the name as identifier of different types of named entities, like a famous work of art, or a famous book, etc. In some cases, however, it can also be the first name. As an example, here is the list of different entries associated to JOHN, first lemma:

```
06043175     John, Gospel_According_to_John
10364758     John, Saint_John, St_John,
             Saint_John_the_Apostle, St_John_the_Apostle,
             John_the_Evangelist,  John_the_Divine
10365110     John, King_John, John_Lackland
```

We have a first mention of JOHN as first member of a synset at 06043175, but then the two following mentions appear one adjacent to the other - thus belonging to the same semantic field (but is this a field at all?). On the other side, we know that when a person name is involved, then the title or the surname is usually needed to address the right person.

This might also not be sufficient, but it is obvious that first (and last) names can be totally ambiguous, without having to be regarded polysemous. Besides, we know that the concept is denoted by the full content of the synset, besides the gloss. And as the content makes it clear, we are here dealing with three totally different referents: one is the Gospel, the other is the Apostle and the third a King. So why use JOHN as first lemma and not the more distinctive second (or third if available) lemma? We find this to be totally misleading from a semantic point of view, because here we are not dealing with polysemous words as was the case with PLANT, but rather with different referential identity. Besides, the word JOHN by itself can have additional uses. Consider for instance the corresponding lower case word "john" which is used with two ambiguous meanings:

```
10076833   whoremaster, whoremonger, john
04274300   toilet, lavatory, lav, can, john, privy, bathroom
```

Here "john" is not the first member of the synset but the difference in meaning is testified again by the distance in terms of offset index values. In these two cases, the choice of lexicographers was not to highlight the polysemy of "john" which appears included in the set but not in first place, and will be assigned the corresponding offset index.

There are only sparse cases of first names as first lemmata in adjacent synsets before reaching the lexicographically marked section of the Noun file where all proper names are collected. Then, the choice to use first/last names as first members of the synset becomes very common in the more restricted list of person names made up of some 3200 entries that start around offset index 110102000. Here are some examples:

```
110102151    Aaron
110102325    Aaron, Henry_Louis_Aaron, Hank_Aaron
110103348    Adam, Robert_Adam
```

```
110103502    Adams, John_Adams, President_Adams, Presi-
             dent_John_Adams
110103654    Adams, John_Quincy_Adams, President_Adams,
             President_John_Quincy_Adams
110103839    Adams, Sam_Adams, Samuel_Adams
110105319    Agrippina, Agrippina_the_Elder
110105487    Agrippina, Agrippina_the_Younger
110109993    Allen, Ethan_Allen
110110169    Allen, Woody_Allen, Allen_Stewart_Konigsberg
110110327    Allen,Gracie_Allen,
             Grace_Ethel_Cecile_Rosalie_Allen, Gracie
110112423    Anderson, Carl_Anderson, Carl_David_Anderson
110112636    Anderson, Marian_Anderson
110112784    Anderson, Maxwell_Anderson
110112893    Anderson,Philip_Anderson,
             Philip_Warren_Anderson,Phil_Anderson
110113110    Anderson, Sherwood_Anderson
```

and the list may continue. In order to cope with this uncouth and unmotivated choice, the algorithm had to be modified: now there would be uncertainty in both files. In the list of LFs, where more than one LF would be associated to each sense and would start with the same word. And in the gloss offset index + synset, where the same first lemma appearing in more than one synset, now has been used to denote a different concept in adjacency. There was no way to use the same automatic approach we used previously. So in order to complete work on the NOUN data file, we have decided to disambiguate each and every synset that needed it: i.e. all those synsets that were associated with more than one LF. After manual modifications, here below is the output and the input for the sequence of adjacent "Anderson":

```
synset(110112423,anderson_NN(x1),['Anderson','Carl_Anderson','Carl_
David_Anderson'])-[united_NN(x1,e6),state_NN(x2,e5),physicist_NN
(x3,e5),discover_VB-[e1,x1,x4],antimatter_NN(x4),in_IN(x4,x5),form_NN
(x5),of_IN(x5,x6),antielectron_NN(x6), call_VB-[e3,x6,e3],positron_NN
(x7,e3)]

synset(110112636,marian_anderson_NN(x1),['Marian_Anderson',
'Anderson'])-[united_NN(x1,e2),state_NN(x2,e2),contralto_NN(x3,e2),
note_VB-[e1,x1],for_IN-[e1,x4],performance_NN(x4),of_IN(x4,x5),
spiritual_NN(x5)]

synset(110112784,anderson_NN(x1),['Anderson','Maxwell_Anderson'])-
[united_NN(x1,e1),state_NN(x2,e1),dramatist_NN(x3,e1)]

synset(110112893,philip_anderson_NN(x1),['Philip_Anderson', 'Ander-
son','Philip_Warren_Anderson','Phil_Anderson'])-[united_NN(x1,e2),
state_NN(x2,e2),physicist_NN(x3,e2),study_VB-[e1,x1,x4],electronic_JJ
(x4),structure_NN(x4),of_IN(x4,x5),magnetic_JJ(x5),disordered_JJ(x5),
system_NN(x5)]

synset(110113110,anderson_NN(x1),['Anderson','Sherwood_Anderson'])-
[united_NN(x1,e2),state_NN(x2,e2),author_NN(x3,e2),works_NN(x4,e2),
be_VB-[e1,x5,x4],frequently_RB(x5,e2),autobiographical_JJ(x5,e2)]
```

which was done after transforming the LFs as follows,

```
lf(anderson_NN(x1),[united_NN(x1),state_NN(x2),physicist_NN(x3),
discover_VB(e1,x1,x4),antimatter_NN(x4),in_IN(x4,x5),form_NN(x5),
of_IN(x5,x6),antielectron_NN(x6),be_VB(e2,x6,e3),call_VB(e3,x8,x6),
positron_NN(x7)]).
lf(marian_anderson_NN(x1),[united_NN(x1),state_NN(x2),contralto_NN
(x3),note_VB(e1,x6,x1),for_IN(e1,x4),performance_NN(x4),of_IN(x4,x5),
spiritual_NN(x5)]).
lf(anderson_NN(x1),[united_NN(x1),state_NN(x2),dramatist_NN(x3)]).
lf(philip_anderson_NN(x1),[united_NN(x1),state_NN(x2),physicist_NN
(x3),study_VB(e1,x1,x4),electronic_JJ(x4),structure_NN(x4),of_IN(x4,
x5),magnetic_JJ(x5),disordered_JJ(x5),system_NN(x5)]).
lf(anderson_NN(x1),[united_NN(x1),state_NN(x2),author_NN(x3),works_NN
(x4),be_VB(e1,x4,x26),frequently_RB(x5),autobiographical_JJ(x5)]).
```

and the offset indices+synsets accordingly,

```
110112423    Anderson, Carl_Anderson, Carl_David_Anderson
110112636    Marian_Anderson, Anderson
110112784    Anderson, Maxwell_Anderson
110112893    Philip_Anderson, Anderson,
             Philip_Warren_Anderson, Phil_Anderson
110113110    Anderson, Sherwood_Anderson
```

5 Conclusion and Future Work

In this paper we presented ongoing work to produce a parser for Logical Forms resources that checks for their consistency, which is basically focussing on the existence of disconnected and ungrounded variables, and tries to correct them. This problem is divided up into two separate processes: one that looks for object variables and tries to connect them to the predicate they depend on. Another process looks for arity of arguments in any predicate formula in order to eliminate unwanted and unneeded variables: these may ensue basically due to the use of a basic lexical structure in presence, however, of omitted arguments. Arguments may be omitted either because they are optional, or because the predicate is used in a passive, intransitivized or ergativized construction. We found an amount of disconnected variables that averages 56% of all LFs, that is 71000 wrong entries over 138000 overall. After running the algorithm for correction which used a lexicon of 7000 verb entries, we managed to correct over 32% of LFs thus reducing the error rate to 24%. We worked then at manually correcting those LFs that are marked as inconsistent by the parser, overall some 4000 entries. We corrected in this way 5.64% of errors that were signaled by the parser. Intervening in this way we discovered new mistakes that are due simply to specific type of structures, containing adjunct structures at verb level. This will require a new effort to count these new mistakes and then manually check the remaining entries. We are also enriching semantically the logical forms, by two types of operations: signaling modifiers' semantic nature as being either restrictive or non-restrictive, then intersective, non-intersective and anti-intersective. But also treating three-place predicates distinguishing closed arguments from predicative arguments.

References

Agerri, R. and Anselmo Peñas (2010) *On the Automatic Generation of Intermediate Logic Form for WordNet glosses*, 11th International Conference on Intelligent Text Processing and Computational Linguistics (Cicling-2010), LNCS Vol. 6008, Springer.

Alshawi, H., Pi-Chuan Chang, M. Ringgaard. (2011) Deterministic Statistical Mapping of Sentences to Underspecified Semantics, in Johan Bos and Stephen Pulman (editors), Proceedings of the 9th International Conference on Computational Semantics, IWCS,15-24.

Bos Johan & Rodolfo Delmonte (eds.), (2008) Semantics in Text Processing (STEP), Research in Computational Semantics, Vol.1, College Publications, London.

Branco, A. (2009) "LogicalFormBanks, the Next Generation of Semantically Annotated Corpora: key issues in construction methodology", In Klopotek, et al., (eds.), Recent Adavnces in Intelligent Information Systems, Warsaw, 3-11.

Bender, E.M. and D.Flickinger (2005) Rapid prototyping of scalable grammars: Towards modularity in extensions to a language-independent core. in Proc. 2nd IJCNLP-05, Jeju Island, Korea.

Bender, E.M., D.Flickinger, and S.Oepen (2002) The Grammar Matrix: An open-source starter-kit for the rapid development of cross-linguistically consistent broad-coverage precision grammars. In J.Carroll et al.(Eds.), Proc. Workshop Grammar Engineering and Evaluation at COLING19, Taipei, Taiwan, 8-14.

Cahill Aoife, Mairead Mccarthy, Michael Burke, Josef Van Genabith, Andy Way (2007) Deriving Quasi-Logical Forms From F-Structures For The Penn Treebank, in Studies in Linguistics and Philosophy, Vol. 83,pp 33-53.

Clark, Peter, Fellbaum, Christiane, and Hobbs, Jerry (2008) Using and Extending WordNet to Support Question Answering. In: Proceedings of the Fourth Global WordNet Conference, eds. A. Tanacs, D. Csendes, V. Vincze, C. Fellbaum and P. Vossen. University of Szeged, Hungary, pp. 111-119.

Copestake, Ann, (2009) Invited Talk: Slacker Semantics: Why Superficiality, Dependency and Avoidance of Commitment can be the Right Way to Go. In: Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), Athens, Greece, pp. 1-9.

Davidson, D. (1967) The logical form of action sentences. In N. Rescher (Ed.), The Logic of Decision and Action, Pittsburgh. University of Pittsburgh Press.

Davidson, D. (1980) Essays on actions and events. Oxford: Clarendon Press.

Harabagiu, S.M., Miller, G.A., Moldovan, D.I.: eXtended WordNet - A Morphologically and Semantically Enhanced Resource (2003) http://xwn.hlt.utdallas.edu, pp. 1-8.

Hobbs, J. (2005) Toward a useful notion of causality for lexical semantics. *Journal of Semantics,* 22:181–209.

Hobbs, J. (2008) Encoding commonsense knowledge. Technical report, USC/ISI. http://www.isi.edu/~hobbs/csk.html.

http1. http://wordnetcode.princeton.edu/standoff-files/wn30-lfs.zip.

http2. http://wordnetcode.princeton.edu/standoff-files/cwn-noun-lfs.txt.

Information Science Institute, University of Southern California: Logical Forms for WordNet 3.0 glosses (2007) http://wordnetcode.princeton.edu/standoff-files/wn30-lfs.zip

Mihalcea, R., and Dan I. Moldovan, (2001) *eXtended WordNet: progress report,* In: Proceedings of NAACL Workshop on WordNet and Other Lexical Resources, Pittsburgh, 95-100.

Miller, George A. & Florentina Hristea, 2006. WordNet Nouns: Classes and Instances, in Computational Linguistics, Vol. 32, No 1, 1-3.

Miller, George A., (editor). 1990. WordNet: An on-line lexical database [Special Issue]. International Journal of Lexicography, 3:235–312.

Moldovan, D. and V. Rus (2001) Explaining answers with extended wordnet. In Proc. ACL'01.

Rathod, Nishit & J. Hobbs (2005) Lftookit. In http://www. isi.edu/rathod /wne /LFToolkit/ index.html, 2005.

Rus, Vasile (2004) A first evaluation of logic form identification systems. In Senseval-3, Association for Computational Linguistics, pp. 37–40.

Schubert, L. and C.Hwang (1993) Episodic logic: A situational logic for NLP. In Peter Aczel, David Israel, Yasuhiro Katagiri, and Stanley Peters, (Eds.), Situation Theory and its Applications, vol.3:303-337.

Predicting word 'predictability' in cloze completion, electroencephalographic and eye movement data

Chris Biemann[1], Steffen Remus[1] and Markus J. Hofmann[2]

[1] Language Technology Group, Comp. Sci. Dept., TU Darmstadt,
Hochschulstr. 10, 64289 Darmstadt, Germany
{biem,remus}@cs.tu-darmstadt.de
[2] General & Biological Psychology, Bergische Universität Wuppertal,
Max-Horkheimer Strasse 20, 42119 Wuppertal, Germany
mhofmann@uni-wuppertal.de

Abstract. Previous neurocognitive approaches to word predictability from sentence context in electroencephalographic (EEG) and eye movement (EM) data relied on cloze completion probability (CCP) data effortly collected from up to 100 human participants. Here we test whether two well-established techniques in computational linguistics can predict these data. Together with baseline predictors of word position and frequency, we found that n-gram language models but not topic models provide an approach to EEG and EM data that is not significantly inferior to the CCP-based predictability data. This is the case for the three corpora we used. Most strikingly, our models accounted for about half of the variance of the CCP-based predictability estimates, thus suggesting that it provides a computational framework to explain the predictability of a word from sentence context. This can help to generalize neurocognitive models to all possible novel word combinations.

1 Introduction

So far, manually collected cloze completion probabilities (CCPs) are typically used for quantifying a word's predictability from sentence context in neurocognitive psychology (Kutas and Hillyard, 1984; Reichle et al., 2003). Here we tackle the question whether the well-understood n-gram language models and Latent Dirichlet Allocation (LDA) topic modeling (Blei et al., 2003) can account for CCPs, as well as whether they can provide an equally well-fitting approach to electroencephalographic (EEG) and eye movement (EM) measures, thus rendering time-consuming CCP procedures unnecessary.

CCPs have been traditionally used to account for N400 responses as an EEG signature of a word's contextual integration into sentence context (Dambacher et al., 2006; Kutas and Hillyard, 1984). Moreover, they were included as the quantification of the theoretical concept of predictability into models of eye movement control (Engbert et al., 2005; Reichle et al., 2003). However, because CCPs are effortly collected from samples of up to 100 participants (Kliegl et al., 2004), they provide a severe challenge to the ability of a model to be generalized across all novel stimuli (Hofmann and Jacobs, 2014), which also prevents their use in technical applications.

To quantify how well computational models of word recognition can account for human performance, Spieler and Balota (1997) proposed that a model should explain variance at the item-level, for instance naming latencies, averaged across a number of participants. Therefore, a predictor variable is fitted to the mean word naming latency y as a function of $y = f(x) = \sum a_n x_n + b + error$ for a number of n predictor variables x that are scaled by a slope factor a, an intercept of b, and an error term. The Pearson correlation coefficient r is calculated, and squared to determine the amount of explained variance r^2. Models with a larger number of n free parameters are more likely to (over-)fit error variance, and thus less free parameters are preferred (e.g., Hofmann and Jacobs, 2014).

While the best cognitive process models can account for 40-50% of variance in behavioral naming data (Perry et al., 2010), neurocognitive data are noisier. The only interactive activation model that gives an amount of explained variance in EEG data (Barber and Kutas, 2007; McClelland and Rumelhart, 1981) was Hofmann et al. (2008), who account for 12% of the N400 variance. Though models of eye movement control use item-level CCPs as predictor variables (Engbert et al., 2005; Reichle et al., 2003), they are rarely investigated in this field (Dambacher and Kliegl, 2007).

While using CCP-data increases the comparability of many studies, the creation of such information is expensive and they only exist for a few languages (Kliegl et al., 2004; Reichle et al., 2003). If it were possible to use (large) natural language corpora and derive the information leveraged from such resources automatically, this would considerably expedite the process of experimentation for under-resourced languages. Comparability would not be compromised when using standard corpora, such as available through Goldhahn et al. (2012) in many languages. However, it is not yet clear what kind of corpus is most appropriate for this enterprise, and whether there are differences in explaining human performance data.

2 Related Work

Taylor (1953) was the first to instruct participants to fill a cloze with an appropriate word. The percentage of participants that fill in the respective word serves as cloze completion probability. For instance, when exposed to the sentence fragment ”He mailed the letter without a ___”, 99% of the participants complete the cloze by ”stamp”, thus CCP equals 0.99 (Bloom and Fischler, 1980). Kliegl et al. (2004) logit-transformed CCPs to obtain *pred = ln(CCP/(1−CCP))*.

Event-related potentials are computed from human EEG data. For the case of the N400, words are often presented word-by-word, and the EEG waves are averaged across a number of participants relative to the event of word presentation. Because brain-electric potentials are labeled by their polarity and latency, the term N400 refers to a negative deflection around 400ms after the presentation of a target word.

After Kutas and Hillyard (1984) discovered the sensitivity of the N400 to cloze completion probabilities, they suggested that it reflects the semantic relationship between a word and the context in which it occurs. However, there are several other factors that determine the amplitude of the N400 (Kutas and Federmeier, 2011, for a

review). For instance, Dambacher et al. (2006) found that word frequency (*freq*), the position of a word in a sentence (*pos*), as well as predictability does affect the N400.

While the eyes remain relatively still during fixations, readers make fitful eye movements called saccades (Radach et al., 2012). When successfully recognizing a word in a stream of forward eye movements, no second saccade to or within the word is required. The time the eyes remain on that word is called single-fixation duration (SFD), which shows a strong correlation to word predictability from sentence context (e.g., Engbert et al., 2005).

3 Methodology

3.1 Human Performance Measures

This study proposes that language models can be benchmarked by item-level performance on three data sets that are openly available in online databases. Predictability was taken from the Potsdam Sentence Corpus 1, first published by Kliegl et al. (2004). The 144 sentences consist of 1138 tokens, available in Appendix A of Dambacher (2009), and the logit-transformed CCP measures of word predictability were retrieved from Ralf Engbert's homepage[1] (Engbert et al., 2005). For instance, in the sentence "Manchmal sagen Opfer vor Gericht nicht die volle Wahrheit" [Before the court, victims tell not always the truth.], the last word has a CCP of 1. N400 amplitudes were taken from the 343 open-class words published in Dambacher and Kliegl (2007). These are available from the Potsdam Mind Research Repository[2]. The EEG data published there are based on a previous study (Dambacher et al., 2006, for method details). The voltage of ten centroparietal electrodes was averaged across 48 artifact-free participants from 300 to 500ms after word presentation for quantifying the N400. SFD are based on the same 343 words from Dambacher and Kliegl (2007), available from the same source URL. Data were included when this word was only fixated for one time, and these SFDs ranged from 50 to 750ms. The SFD was averaged across up to 125 German native speakers (Dambacher and Kliegl, 2007).

3.2 N-gram Language and LDA Topic Models

Language models are based on a probabilistic model of language. The resulting probabilities can be used to pick the most fluent of several alternatives e.g. in machine translation or speech recognition. Word **n-gram models** are defined by a Markov chain of order $n-1$, where the probability of the following word only depends on previous $n-1$ words. The probability distribution of the vocabulary, given a history of $n-1$ words, is estimated based on n-gram counts from (large) natural language corpora. There exist a range of n-gram language models (see for example Chapter 3

[1] http://mbd.unipotsdam.de/EngbertLab/Software.html
[2] http://read.psych.unipotsdam.de

in Manning and Schütze, 1999). Here, we use a Kneser and Ney (1995) 5-gram model[3]. For each word in the sequence, the language model computes a probability $p \in]0; 1[$. We use the logarithm *log(p)* of this probability as predictor. We used all words in their full form, i.e. did not filter for specific word classes and did not perform lemmatization. N-gram language models are known to model local syntactic structure very well. Since only n-gram models use the most recent history for predicting the next token, they fail to account for long-range phenomena and semantic coherence, cf. (Biemann et al., 2012).

Latent Dirichlet Allocation (LDA) topic models (Blei et al., 2003) are generative probabilistic models representing documents as a mixture of a fixed number of N topics, which are defined as unigram probability distributions over the vocabulary. Through a sampling process like Gibbs sampling, topic distributions are inferred. Words frequently co-occurring in the same documents receive a high probability in the same topics. When sampling the topic distribution for a sequence of text, each word is randomly assigned to a topic according to the document-topic distribution and the topic-word distribution. We use Phan and Nguyen's (2007) GibbsLDA implementation for training an LDA model with 200 topics (default values for $\alpha = 0.25$ and $\beta = 0.001$) on a background corpus. Words occurring in too many documents (a.k.a. stopwords) were removed from the LDA vocabulary. Then, we repeatedly sample the topic assignments (cf. Riedl and Biemann, 2012) on the input sentence and retain the most frequently assigned three topics per word. As predictor for the current open class word in the sequence, we count the number of previous open class words in the sequence, which have at least one topic in common with the current word. Intuitively, this measure should capture the amount of semantic coherence with the previous words in the sequence. I.e. for a sequence like "The dwarf was avoiding the ____", we'd expect a score of 1 for "elves" for their topical similarity to "dwarf" (provided that there is sufficient support of dwarves and elves in the background corpus), whereas we expect a score of 0 for "rain". Parameters of this procedure were determined in preliminary experiments. We hypothesized that topic models account for the semantic aspects missing in n-gram models. While Bayesian topic models are probably the most widespread approach to semantics in psychology (e.g., Griffiths et al., 2007), latent semantic analysis (LSA) is not applicable in our setting (Landauer and Dumais, 1997): we use the capability of LDA to account for yet unseen documents, whereas LSA assumes a fixed vocabulary and document space at model construction time. In further experiments, we also used collocation statistics to predict semantically expected items, but we obtained no correlation with human data.

4 Experiment Setup

Engbert et al. (2005)'s data are organized in 144 short German sentences with an average length of 7.9 tokens, and provide features, such as *freq* as corpus frequency in occurrences per million (Baayen et al., 1995), *pos*, and *pred*. We test whether two

[3] https://code.google.com/p/berkeleylm/

corpus-based predictors can account for predictability, and compare the capability of both approaches in accounting for EEG and EM data. For training n-gram and topic models, we used three different corpora differing in size and covering different aspects of language. Further, the units for computing topic models differ in size.

NEWS: A large corpus of German online newswire from 2009 as collected by LCC (Goldhahn et al., 2012) of 3.4 million documents / 30 million sentences / 540 million tokens. This corpus is not balanced, i.e. important events in the news are covered better than other themes. The topic model was trained on the article level.

WIKI: A recent German Wikipedia dump of 114,000 articles / 7.7 million sentences / 180 million tokens. This corpus is rather balanced, as concepts or entities are described in a single article each, independent of their popularity, and spans all sorts of topics. The topic model was trained on the article level.

SUB German subtitles from a recent dump of opensubtitles.org, containing 7420 movies / 7.3 million utterances / 54 million tokens. While this corpus is much smaller than the others, it is closer to a colloquial use of language. Brysbaert et al. (2011) showed that word frequency measures of subtitles provide numerically greater correlations with word recognition speed than larger corpora of written language. The topic model was trained on the movie level.

Pearson's product-moment correlation coefficient was calculated (e.g. Coolican, 2010, p. 293), and squared for the N = 1138 predictability scores (Engbert et al., 2005) or N = 343 N400 amplitudes or SFD (Dambacher and Kliegl, 2007). To address overfitting, we randomly split the material in two halves, and test how much variance can be reproducibly predicted on two subsets of 569 items. For N400 amplitude and SFD, we used the full set, because one half was too small for reproducible predictions.

5 Results

5.1 Predictability results

In the first series of results, we examine the correlation of manually obtained predictability with corpus-based methods. High correlations would indicate that predictability could be replaced by automatic methods. As a set of baseline predictors, we use *pos* and *freq*, which explains 0.243 / 0.288 of the variance for the first respectively the second half of the dataset. We report results in Table 1 for all single corpus-based predictors alone and in combination with the baseline, all combinations of the baseline with n-grams and topics from the same corpus.

predictors	NEWS	WIKI	SUB
n-gram alone	.262/.294	.226/.253	.268/.272
topic alone	.024/.037	.029/.022	.014/.012
base+n-gram	.462/.490	**.462/.490**	**.448/.459**
base+topic	.252/.307	.254/.296	.244/.289
base+both	**.481/.516**	.445/.473	**.449/.461**

Table 1. r^2 explained variance of predictability, given for two folds of the data set, for various combinations of baseline and corpus-based predictors.

It is apparent that the n-gram predictor alone reaches r^2 levels comparable to the baseline, whereas the topic model alone explains hardly any variance. Combining the baseline with the n-gram predictor achieves the best fitting to predictability for the WIKI and SUB corpora. Combining the baseline with topics shows small improvements for NEWS and WIKI (see Figure 1).

The best overall performance based on a single corpus is achieved with combining the baseline with n-grams and topics from the NEWS corpus. This confirms a generally accepted hypothesis that larger training data trumps smaller, more focused training data, see e.g. (Banko and Brill, 2001) and others. We also fitted a model based on all corpus-based predictors from all corpora, which achieved the overall highest r^2= 0.532 / 0.547 . From these experiments it becomes clear that predictability can largely be explained by a combination positional and frequency features combined with a word n-gram language model. Different corpora capture slightly different aspects of predictability, which is reflected by the improvements when combining predictors from all three corpora. The topic model-based predictor only shows a negligible influence.

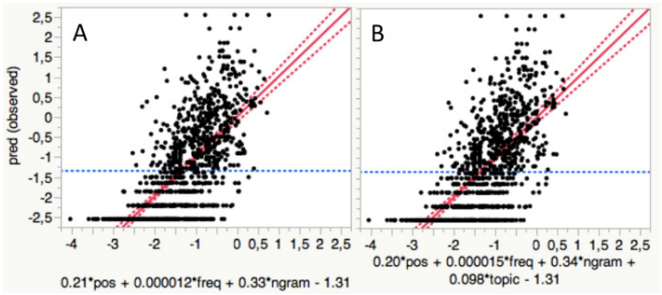


Fig. 1. Prediction models exemplified for the NEWS corpus in the x-axes and the N = 1138 predictability scores on the y-axes. A) shows the prediction by baseline + n-gram (r^2=0.475), and in B) a topic-predictor was added (r^2=0.481). Fisher's r-to-z test revealed that there is no significant difference in explained variance (P=0.82)

5.2 N400 and SFD results

For modeling **N400**, we have even more combinations at our disposal since we can combine the baseline with predictability as given in the dataset, with corpus-based measures, and with both. We evaluate on all 343 data points for N400 amplitude fitting. Without using corpus-based predictors, the baseline predicts a mere 0.032 of variance, predictability alone explains 0.192 of variance, and their combination explains 0.193 – i.e. the baseline is almost entirely subsumed by predictability.

Fig. 2 lists the results for N400 amplitude modeling with corpus-based predictors. Again, the n-gram model is the best corpus-based predictor, and fares best when

trained on the NEWS corpus, confirming the result that corpus size is the major factor for n-gram model quality. For the N400 experiments, the difference between the larger corpora (NEWS, WIKI) and the smaller corpus (SUB) is more pronounced. Again, the topic predictor fails to show a major influence for explaining N400 amplitude variance. The best combination without predictability, with a score of $r^2 = 0.182$, comes close to the performance of predictability alone.

predictors	NEWS	WIKI	SUB
n-gram alone	0.141	0.140	0.126
topic alone	0.022	0.021	0.006*
n-gram+topic	0.170	0.166	0.131
base+n-gram	0.161	0.153	0.135
base+topic	0.051	0.050	0.036
bas+n-gram+topic	**0.182**	0.172	0.137
base+pred+n-gram	**0.223**	**0.226**	0.206
base+pred+topic	0.194	0.193	0.193
base+pred+both	**0.228**	**0.229**	0.206

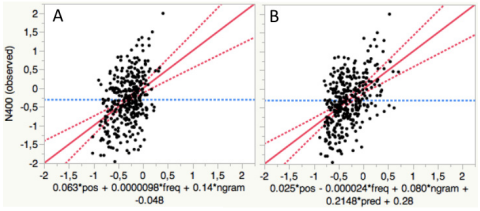


Fig. 2. Left: r^2 explained variance of N400 amplitude, for various combinations of baseline, predictability and corpus-based predictors. * marks statistically independent predictors of N400 ($p > 0.05$). Right: Two prediction models exemplified for the NEWS corpus in the x-axes and the N = 343 N400 amplitudes on the y-axes. A) shows the prediction by baseline + n-gram, and in B) predictability was added. Fisher's r-to-z test revealed that there is no significant difference in explained variance (P=0.25)

The experiments with predictability as an additional predictor confirm the results from the previous section: n-grams + baseline and predictability capture slightly different aspects of human reading performance, thus their combination explains about 3% more variance than predictability alone. This difference, however, is not statistically reliable (see Figure 2). Differences between the two large corpora are negligible, and so is the influence of the topic-based predictor.

Finally, we examine the corpus-based predictors for modeling the mean **single fixations duration** for 343 words. For this target, the *pos+freq* baseline explains $r^2 = 0.021$, whereas predictability, alone or combined with the baseline, explains $r^2 = 0.184$.

predictors	NEWS	WIKI	SUB
n-gram alone	0.225	0.140	0.126
topic alone	0.006*	0.006*	0.006*
n-gram+topic	0.231	0.223	0.226
base+n-gram	**0.239**	0.226	0.226
base+topic	0.023	0.024	0.029
bas+n-gram+topic	**0.242**	0.230	0.229
base+pred+n-gram	**0.273**	**0.274**	0.258
base+pred+topic	0.188	0.184	0.184
base+pred+both	**0.273**	**0.274**	0.259

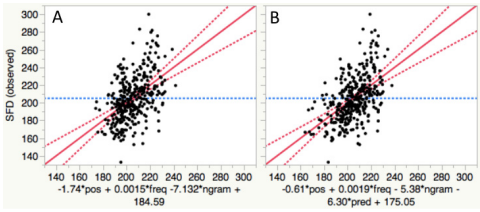


Fig. 3. Left: r^2 explained variance of single-fixation duration, for various combinations of baseline, predictability and corpus-based predictors. * marks statistically independent predictors of SFD ($P > 0.05$). Right: Two prediction models exemplified

for the NEWS corpus in the x-axes and the N = 343 SFD on the y-axes. A) shows the prediction by baseline + n-gram, and in B) predictability was added. Fisher's r-to-z test revealed that there is no significant difference in explained variance (P=0.56)

The experiments confirm the utility of n-gram models in accounting for eye movement data. Adding predictability did not lead to a significant increase of variance explained (see Fig. 3). In addition, the n-gram model alone explains more variance than predictability – however, the difference is not significant.

For SFD, corpus size does not seem to be a major influencing factor, as results are comparable across corpora, however with the largest corpus (NEWS) still yielding the best modeling results overall in absence of the predictability predictor. For SFD, topic models seem entirely uncorrelated.

And again, the experiments confirm that n-gram models and predictability capture similar, but slightly different aspects, since their combination yields another improvement, explaining $r^2 = 0.273$ overall.

6 Conclusion

We have examined the utility of two corpus-based predictors to account for word predictability from sentence context, as well as the EEG signals and EM-based reading performance elicited by it. Our hypothesis was that word n-gram models and topic models would account for the predictability of a token, given the preceding tokens in the sentence, as perceived by humans. Our hypothesis was at least partially confirmed: n-gram models, sometimes in combination with a frequency-based and positional baseline, are highly correlated with human predictability scores and in fact explain variance of human reading performance to an extent comparable to predictability – slightly less on N400 but slightly more on SFD.

Topic models on the other hand, at least in the particular way we used them here, failed to show a major influence on modeling human reading performance. This might be related to the fact that the sentence scope in the data set is rather short so that most "priming" effects can already be captured by our 5-gram model – topic models usually perform well on the level of documents, not single sentences.

Can we now safely replace human predictability scores with n-gram statistics? Given the high correlation between predictability and the combination of n-grams with frequency and positional information, and given that n-gram-based predictors achieve similar levels of explained variance than predictability, the answer seems to be positive. However, though our corpus-based approaches explain most of the variance that by manually collected CCP scores also account for, adding predictability always accounts for more variance – though this difference is not significant (see Figures 1-3). It is yet an open question, whether additional corpus-based predictors, be it topic models or something else, could entirely explain the prediction power of human CCP data for tasks like N400 amplitude and SFD modeling.

While n-gram models together with word frequency and position captured about half of the predictability variance, and most of the N400 and SFD variance elicited by it, we propose that it can be used to replace tediously collected CCPs. This not only

saves a lot of pre-experimental work, but it also opens the possibility to apply (neuro-) cognitive models in technical applications. For instance, n-gram models can be used to generalize computational models of eye movement control to novel sentences (Engbert et al., 2005; Reichle et al., 2003).

In the end, this will also improve our understanding of the cognitive processes underlying EM and EEG measures. While both of these are not as well understood as human CCP performance, predictability provided a great step towards understanding the determinants of neurocognitive prediction processes. If we can compute the determinants of N400 and SFDs from a corpus of sentences, however, we can computationally define these cognitive processes rather than using a better-understood performance (CCP) to account for other human performance (N400, SFD).

Baayen (2010) proposed word frequency to be a collector variable often subsuming other highly correlated variables. We found that adding n-grams to the baseline of pos and freq doubled the explained variance in CCP-based predictability scores. This suggests that the sentence level can unfold the cognitive processes previously ascribed to word frequency. The doubling of explained variance suggests still unexploited sources of human variance to be explained by neurocognitive simulation models, which quantify the contextual constraints imposed by position-sensitive predictions of a sentence's words (e.g. Hofmann & Jacobs, 2014; Kutas & Federmeier, 2011).

Much as for computational models of word recognition, the amount of explained item-level variance can serve as a benchmark for language models. Such a common benchmark facilitates the comparison of differential computational models. Thus, for instance, we would not only know that Frank et al. (2013)'s novel language model can account for the N400, but the common benchmark of explained variance could be easily compared to any novel approach – for instance by assessing whether one measure is significantly better than another one for the purpose of modeling.

Acknowledgments

The "Deutsche Forschungsgemeinschaft" (MJH; HO 5139/2-1), the German Institute for Educational Research in the Knowledge Discovery in Scientific Literature (SR) program and the LOEWE center for Digital Humanities (CB) supported this work.

References

H. R. Baayen, R. Piepenbrock, and L. Gulikers (1995) The CELEX Lexical Database. Release 2 (CD-ROM). LDC, University of Pennsylvania, Philadelphia.

H.R. Baayen (2010). Demythologizing the word frequency effect: A discriminative learning perspective. *The Mental Lexicon*, 5(3):436-461.

M. Banko and E. Brill (2001) Scaling to very very large corpora for natural language disambiguation. *Proc. ACL '01*, pp. 26–33, Toulouse, France.

H. A. Barber and M. Kutas (2007) Interplay between computational models and cognitive electrophysiology in visual word recognition. *Brain Res. Rev.*, 53(1):98–123.

C. Biemann, S. Roos, K. Weihe (2012) Quantifying semantics using complex network analysis. *Proc. COLING 2012*, pp. 263–278, Mumbai, India.

D. M. Blei, A. Y. Ng, M. I. Jordan (2003) Latent Dirichlet Allocation. *Journal of Machine Learning Research*, 3:993–1022.

P. A. Bloom and I. Fischler (1980) Completion norms for 329 sentence contexts. *Memory & cognition*, 8(6):631–642.

M. Brysbaert, M. Buchmeier, M. Conrad, A. M. Jacobs, J. Bölte, and A. Böhl (2011) A Review of Recent Developments and Implications for the Choice of Frequency Estimates in German. *Experimental psychology*, 58:412–424.

H. Coolican (2010) Research Methods and Statistics in Psychology. *Hodder & Stoughton*.

M. Dambacher and R. Kliegl (2007) Synchronizing Timelines: Relations between fixation durations and N400 amplitudes during sentence reading. *Brain research*, 1155:147–162.

M. Dambacher, R. Kliegl, M. J. Hofmann, A. M. Jacobs. (2006) Frequency and predictability effects on event-related potentials during reading. *Brain research*, 1084(1):89–103.

M. Dambacher. 2009. Bottom-up and top-down processes in reading. *Universitätsverlag Potsdam*, Potsdam.

R. Engbert, A. Nuthmann, E. M. Richter, R. Kliegl (2005) SWIFT: a dynamical model of saccade generation during reading. *Psychological review*, 112(4):777–813.

S. L. Frank, G. Galli, and G. Vigliocco (2013) Word surprisal predicts N400 amplitude during reading. *Proc. ACL-2013,* pp. 878–883, Sofia, Bulgaria.

D. Goldhahn, T. Eckart, U. Quasthoff (2012) Building large monolingual dictionaries at the Leipzig Corpora Collection: From 100 to 200 languages. *Proc. LREC 2012*, Istanbul, Turkey.

T. L. Griffiths, M. Steyvers, and J. B. Tenenbaum (2007) Topics in Semantic Representation. *Psychological review*, 114(2):211–244.

M. J. Hofmann and A. M. Jacobs (2014) Interactive activation and competition models and semantic context: From behavioral to brain data. *Neuroscience and biobehav. rev.s*, 46:85–104.

M. J. Hofmann, S. Tamm, M. M. Braun, M. Dambacher, A. Hahne, and A. M. Jacobs (2008) Conflict monitoring engages the mediofrontal cortex during nonword processing. *Neuroreport*, 19(1):25–9.

R. Kliegl, E. Grabner, M. Rolfs, and R. Engbert (2004) Length, frequency, and predictability effects of words on eye movements in reading. *Europ. Journal of Cog. Psy.*, 16(12):262–284.

R. Kneser and H. Ney (1995) Improved backing-off for m-gram language modeling. *Proc. IEEE Int'l Conf. on Acoustics, Speech and Signal Processing*, pp. 181–184, Detroit, Michigan.

M. Kutas and K. D. Federmeier (2011) Thirty years and counting: finding meaning in the N400 component of the event-related brain potential (ERP). *Ann. Rev. of Psychology*, 62:621–47.

M. Kutas and S. A. Hillyard (1984) Brain potentials during reading reflect word expectancy and semantic association. *Nature*, 307(5947):161–3.

T. K. Landauer and S. T. Dumais (1997) A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104(2):211–240.

C. D. Manning and H. Schütze (1999) Foundations of Statistical Natural Language Processing. *MIT Press*, Cambridge, MA, USA.

J. L. McClelland and D. E. Rumelhart (1981) An Interactive Activation Model of Context Effects in Letter Perception: Part 1. *Psychological Review*, 5:375–407.

C. Perry, J. C. Ziegler, and M. Zorzi (2010) Beyond single syllables: large-scale modeling of reading aloud with the Connectionist Dual Process (CDP++) model. *Cognitive Psychology*, 61(2):106–51.

X-H. Phan and C-T. Nguyen (2007) GibbsLDA++: A C/C++ implementation of latent Dirichlet allocation (LDA). http://jgibblda.sourceforge.net/.

R. Radach, T. Günther, and L. Huestegge (2012) Blickbewegungen beim Lesen, Leseentwicklung und Legasthenie. *Lernen und Lernstoerungen*, 1(3):185–204.

E. D. Reichle, K. Rayner, and A. Pollatsek (2003) The E-Z reader model of eye-movement control in reading: comparisons to other models. *The Behavioral and brain sciences*, 26(4):445–76; discussion 477–526.

M. Riedl and C. Biemann (2012) Sweeping through the topic space: Bad luck? roll again! *Proc. ROBUS-UNSUP 2012*, Avignon, France.

D. H. Spieler and D. A. Balota (1997) Bringing Computational Models of Word Naming Down to the Item Level. *Psychological Science*, 8(6):411–416

W. L. Taylor (1953) "Cloze" procedure: A new tool for measuring readability. *Journalism Quarterly*, 30:415.

Deterministic Choices in a Data-driven Parser

Sardar Jaf[1], Allan Ramsay[1]

[1]The University of Manchester, Faculty of Engineering and Physical Sciences, School of Computer Science, Manchester, United Kingdom
{sardar.jaf, allan.ramsay}@manchester.ac.uk

Abstract. Data-driven parsers rely on recommendations from parse models, which are generated from a set of training data using a machine learning classifier, to perform parse operations. However, in some cases a parse model cannot recommend a parse action to a parser unless it learns from the training data what parse action(s) to take in every possible situation. Therefore, it will be hard for a parser to make an informed decision as to what parse operation to perform when a parse model recommends no/several parse actions to a parser. Here we examine the effect of various deterministic choices on a data-driven parser when it is presented with no/several recommendation from a parse model.

1 Introduction

One of the main components of a data-driven parser is a parse model, which recommends parse operations to a parser when processing sentences. It is not guaranteed that a parse model can cover every possible situation during parsing and hence it may be unable to recommend a parse operation or it may recommend several operations in a given situation. Therefore, when a parse model recommends no/several operations to a parser, it will be hard for the parser to determine what operation to perform. In Section 3 we will describe a basic shift-reduce parser while in Section 4 we will describe our parser. In Section 6 we will identify several deterministic choices that a data-driven shift-reduce parser may take. We will examine the effect of these deterministic choices on the parsing performance in terms of efficiency and accuracy. In Section 8.1, we will examine the effect of various deterministic choices when running our parser deterministically, and in Section 8.2 we will examine the effect of the deterministic choices on our parse when running it non-deterministically.

2 Dataset

We have used the Penn Arabic Treebank (PATB) (Maamouri and Bies, 2004) part 1 version 3 for training and testing our dependency data-driven parser, which is a re-

implementation of the arc-standard version of MaltParser (Nivre et al., 2010; Kuhlmann and Nivre, 2010; Nivre et al., 2006). We have converted the phrase structure trees of the PATB to dependency structure trees using the standard conversion algorithm for transforming phrase structure trees to dependency trees, as described by Xia and Palmer (2001). In order to perform a 5-fold validation, we have systematically generated five sets of testing data and five sets of training data from the treebank, where the testing data is not part of the training data. The training data contains approximately 3853 sentences. The average length of sentences is 29 words and the total number of testing sentences in each fold is about 970 sentences.

3 A Shift-reduce Parser

A basic shift-reduce parsing algorithm performs one out of three operations at any parse transitions: SHIFT, LEFT-ARC or RIGHT-ARC. These operations are applied to a queue of words which have not yet been looked at and a stack of words which have been inspected but have not yet been assigned a syntactic role.

The SHIFT operation moves the head of the queue to the top of the stack. The LEFT-ARC and RIGHT-ARC operations establish head-dependent relations (in dependency parsing) between the head item of the queue and the top item on the stack. The LEFT-ARC and the RIGHT-ARC operations are applied to one node in a queue of input strings and one node on the stack. The LEFT-ARC operation makes the first node in the queue the parent of the top node on the stack while the RIGHT-ARC operation makes the top node on the stack the parent of the first node in the queue and rolls back the item on the top of the stack to the queue.

Our parser implementation is similar to the arc-standard algorithm of MaltParser (Kuhlmann and Nivre, 2010), which takes a deterministic approach to parsing natural language text where a support vector machine (SVM) (Chang and Lin, 2001) classifier is used for learning parse operations from a dependency treebank. The classifier helps the parser to predict the most likely correct parse operation when it is presented with a non-deterministic choice between multiple parse operations. As Nivre (2008) states, in this kind of implementation the parser derives a single parse analysis by incrementally selecting a parse operation, which makes the parsing process very simple and efficient. Moreover, by using an appropriate classifier, a good parsing accuracy is achievable (Nivre, 2008, p. 514).

The original arc-standard algorithm uses a deterministic approach to parsing natural language texts. The parser follows suggestions made by a parse model to perform a specific parse action (SHIFT, LEFT-ARC, or RIGHT-ARC) at each parse step. Performing the wrong parse action at a particular step during parsing will have a knock on effect on subsequent parsing steps. Hence, the error propagation can be substantial. Using a non-deterministic approach, where the parser is presented with multiple actions to take, allows the parser to recover from a previous mistake if this is subsequently identified.

4 DNDParser

Our parser contrasts with MaltParser in the way it is non-deterministic but with some deterministic features. We will call our parser DNDParser, which is short for deterministic and non-deterministic dependency data-driven parser. At each parse step, we generate a state for SHIFT, LEFT-ARC, and RIGHT-ARC, and we will assign different scores to each state. The score of each state is computed by using two different scores: (i) a score that is based on the recommendation made by the parse model. For example, we give a score of 1 for a SHIFT operation if it is recommended by the parse model, otherwise we give it a score of 0 (and the same applies to LEFT-ARC and RIGHT-ARC). (ii) We add the score from (i) to the score of the current state (which is the state that the new parse state is generated from). The sum of these two scores is assigned to the newly generated parse state(s). We can rank a collection of parse states by using their scores and then process the state with the highest score, which we consider the most plausible state. The various states generated by our parser is described in the following section.

5 Assigning Scores to Parse States

We extend the LEFT-ARC and RIGHT-ARC operations of the shift-reduce algorithm to allow more variations of the reduce operations, such as LEFT-ARC(*n*) and RIGHT-ARC(*n*) where *n* is any positive numbers. In this way, our parser generates one or more parse states from a given state based on following situations:

- If the queue consists of one or more items and the stack is empty then the parser produces one state by performing SHIFT. For example, if the queue consists of items such as [1, 2, 3, 4] and an empty stack such as [] then the parser cannot recommend LEFT-ARC(*n*) or RIGHT-ARC(*n*) because these two operations require an item on the stack to be made the parent or the daughter of the head of the queue respectively
- If the queue consists of one or more items such as [2, 3, 4] and the stack consists of one item only such as [1], then there are three possible moves: SHIFT, LEFT-ARC(1), and RIGHT-ARC(1). However, the parse model, which is based on a classification algorithm, will recommend only one operation (SHIFT, LEFT-ARC(1), or RIGHT-ARC(1)). Hence, in this kind of state our parser generates three states but only one state will be given a positive score, which is based on recommendation of the parse model.
- If the queue consists of one or more items such as [3, 4] and the stack consists of more than one item such as [2, 1], then our parser may generate more than three states because it checks for relations between the head of the queue and any items on the stack; i.e., states

that are generated by LEFT-ARC(n+1) and RIGHT-ARC(n+1). This approach is a generalisation of proposals by Kuhlmann and Nivre (2010) and Attardi (2006).

We store the states with various scores in an agenda sorted based on their scores, and the state with the highest score is explored by the parser.

6 Classification-driven Deterministic Parsing

During some parse transitions, DNDParser may be forced to make deterministic decisions. As explained in the previous section, if the parser is presented with a state that has one or more items on the queue but an empty stack then it will produce one state by performing SHIFT. For example, having a queue with [1, 2, 3, 4] and an empty stack [] then the parser cannot recommend LEFT-ARC or RIGHT-ARC because both of these two operations requires an item from the stack to be made the parent or the daughter of the head of the queue.

Having one or more items on the queue and one item on the stack the parser produces three states, namely: SHIFT, LEFT-ARC, and RIGHT-ARC. In this kind of situation, the parsing model recommends only one operation where we give it a positive score so that the parser can then explore the recommended operation. However, it is possible that the parse model may not recommend any operations if it is presented with a situation that has never seen it during training. This is possible because the classifier may not learn what action to take in every situation the parser encounters during the testing phase. For example, in Fig. 1 we assume that the parse model did not recommend any operation, where all three operations receive a score of 0, and thus they will all have equal scores (which is the score inherited from the original state).

States	Operations	Queue	Stack	Score
Current state	-	[2, 3, 4]	[1]	0
New states	SHIFT	[3, 4]	[2, 1]	0
	RIGHT-ARC(1)	[1, 3, 4]	[]	0
	LEFT-ARC(1)	[2, 3, 4]	[]	0

Fig. 1. Generating three parse states from one state

In this kind of situation, it is not clear which operation the parser should explore first, LEFT-ARC(1), RIGHT-ARC(1) or SHIFT. There are six different deterministic strategies (order-of-preference) we can give to the parser as to which operation it should explore first, those are:

1. SHIFT-LEFT-ARC-RIGHT-ARC

2. SHIFT-RIGHT-ARC-LEFT-ARC
3. LEFT-ARC-SHIFT-RIGHT-ARC
4. LEFT-ARC-RIGHT-ARC-SHIFT
5. RIGHT-ARC-SHIFT-LEFT-ARC
6. RIGHT-ARC-LEFT-ARC-SHIFT

Furthermore, in situations where the parser is presented with a state that has one or more items on the queue and more than one items on the stack, the parser can then generate more than three states because it checks for relations between the head of the queue and any items on the stack; i.e., states that are generated by LEFT-ARC(n+1) and RIGHT-ARC(n+1). In this kind of situation, it is possible that two or more operations may be recommended by the parse model, where two or more states receive positive scores. For example, in Fig. 2 where the parsing rules suggested LEFT-ARC(1) (making 3 from the queue the parent of 2 on the stack) and also LEFT-ARC(2) (making 3 the head of the queue the parent of 1 from the stack) they are both given a score of 1.

States	Operations	Queue	Stack	Tree	Score
Current state	-	[3, 4]	[2, 1]	-	0
New states	SHIFT	[4]	[3, 2, 1]	-	0
	RIGHT-ARC(1)	[2, 4]	[1]	(2>3)	0
	RIGHT-ARC(2)	[1, 2, 4]	[]	(1>3)	0
	LEFT-ARC(1)	[3, 4]	[1]	(3>2)	1
	LEFT-ARC(2)	[2, 3, 4]	[]	(3>1)	1

Fig. 2. Generating more than three parse states from one state

In this kind of exemplified situation we may deterministically choose to perform LEFT-ARC(1) instead of LEFT-ARC(2), by giving more priority to reduce operations that involve two items that are closer to each other. Alternatively, we may deterministically choose LEFT-ARC(2), by giving priority to reduce operations that involve two items that are further away from each other. This leads to another two different deterministic choices, which are:

1. furthest-item-first: this operation involves making relations between the head of the queue and an item that is furthest away from it on the stack.
2. closest-item-first: this operation involves making relations between the head of the queue and an item on the stack that is closest to it on the stack.

We can run the parser deterministically by allowing it to accept the first terminal state that it produces, which is a state where there are no possible actions for the parser to take (i.e. if the queue is empty).

7 Classification-driven Non-deterministic Parsing

Running our parser completely deterministic, then we allow it to accept the first terminal state it produces (whether a well-formed tree is produced); i.e., when the queue becomes empty because processing of all the words in it is performed by removing queue items on to the stack. If we run the parser non-deterministically, we allow it to explore the alternative states that remain on the agenda if the first terminal state is not well-formed; i.e., where the stack has more than one item on it, which means that some words did not receive a parent and hence a complete parse tree is not produced. This means that the parser rolls back to the previous highest scored state on the agenda and explores it until a state is generated whereby the stack contains one item and a complete parse tree is generated.

8 Evaluation

In this section, we will present our evaluation of the deterministic and non-deterministic versions of DNDParser. We show three different parsing accuracy measures, those are: (i) Labelled Attachment Scores (LAS), which is the percentage of the correct dependency relations with the correct labels of the dependency relations (DEPREL) between tokens; (ii) Unlabelled Attachment Score (UAS), which is the percentage of correct dependency relation (i.e., the percentage of tokens with correct heads) regardless of the DEPREL; and (iii) Labelled Accuracy (LA) which is the percentage of tokens with the correct dependency label. The efficiency of the parser is also presented, which is amount of time in seconds the parser consumes for establishing a dependency relation between two words.

8.1 Deterministic Parser Evaluation with Various Deterministic Choices

In this section we will evaluate DNDParser by running it completely deterministic. In deterministic mode, the parser accepts the first terminal state it produces regardless of whether the state contains a complete parse tree for a given sentence. Moreover, we present results for the various deterministic strategies, which we outlined in Section 6. We can observe from Table 1 that from the six deterministic order-of-preferences, the LEFT-ARC-SHIFT-RIGHT-ARC strategy produces the highest parsing accuracy.

We can also observe that the LEFT-ARC-SHIFT-RIGHT-ARC order-of-preference produces higher parsing accuracy when combined with the furthest-item-first reduction strategy than when it is combined with the closest-item-first reduction strategy. However, combining the LEFT-ARC-SHIFT-RIGHT-ARC order-of-preference with the furthest-item-first reduction strategy degrades the parsing efficiency by about 7% compared with when it is combined with the closest-item-first reduction strategy.

Table 1. Deterministic parsing evaluation

Furthest-item-first reduction				
Strategy	UAS (%)	LAS (%)	LA (%)	Efficiency
LEFT-ARC-SHIFT-RIGHT-ARC	**72.48**	**70.63**	**93.6**	**0.062**
SHIFT-LEFT-ARC-RIGHT-ARC	59.77	58.12	72.1	0.047
SHIFT-RIGHT-ARC-LEFT-ARC	59.41	57.76	71.7	0.067
RIGHT-ARC-LEFT-ARC-SHIFT	53.67	52.25	87.8	0.043
LEFT-ARC-RIGHT-ARC-SHIFT	53.67	52.25	87.8	0.042
RIGHT-ARC-SHIFT-LEFT-ARC	53.27	52.15	87.8	0.041

Closest-item-first reduction				
Strategy	USA (%)	LAS (%)	LA (%)	Efficiency
LEFT-ARC-SHIFT-RIGHT-ARC	**66.46**	**64.72**	**92.6**	**0.058**
SHIFT-LEFT-ARC-RIGHT-ARC	59.76	58.05	73.4	0.035
SHIFT-RIGHT-ARC-LEFT-ARC	59.58	57.87	73.3	0.41
RIGHT-ARC-SHIFT-LEFT-ARC	52.62	51.18	87.7	0.037
RIGHT-ARC-LEFT-ARC-SHIFT	51.35	49.96	87.7	0.030
LEFT-ARC-RIGHT-ARC-SHIFT	51.15	49.26	87.5	0.032

8.2 Non-deterministic Parser Evaluation with Various Deterministic Choices

In this section we will evaluate our parser by running it non-deterministically. In this mode, the parser explores other states until it finds a well-formed terminal state, which is a state where the stack contains one item and a complete parse tree is generated. We run the parser in this mode by integrating various deterministic strategies that we outlined in Section 6. We can note from Table 2 that from the six deterministic order-of-preferences (see Section 6 for more detail), the SHIFT-LEFT-ARC-RIGHT-ARC order-of-preference produces the highest parsing accuracy. We can also observe that the SHIFT-LEFT-ARC-RIGHT-ARC order-of-preference produces higher parsing accuracy when combined with the furthest-item-first reduction strategy than when it is combined with the closest-item-first reduction strategy. However, combining the SHIFT-LEFT-ARC-RIGHT-ARC strategy with any of the two strategies (furthest-item-first reduction or closest-item-first reduction) the speed of the parse is not largely affected (about 2.4%).

It appears that using different settings affects the performance of the parser greatly. From the experiments conducted in this section, and the previous section, it is apparent that running the parser non-deterministically with SHIFT-LEFT-ARC-RIGHT-ARC order-of-preference and using the furthest-item-first reduction strategy produces the best parsing performance.

Table 2. Non-deterministic parsing evaluation with different deterministic choices

Furthest-item-first reduction				
Strategy	UAS (%)	LAS (%)	LA (%)	Efficiency
SHIFT-LEFT-ARC-RIGHT-ARC	**74.5**	**71.0**	**93.6**	**0.081**
LEFT-ARC-SHIFT-RIGHT-ARC	72.6	70.7	92.0	0.072
SHIFT-RIGHT-ARC-LEFT-ARC	57.5	55.8	88.1	0.074
RIGHT-ARC-LEFT-ARC-SHIFT	53.6	52.2	87.9	0.060
LEFT-ARC-RIGHT-ARC-SHIFT	53.6	52.2	87.9	0.059
RIGHT-ARC-SHIFT-LEFT-ARC	53.6	52.2	87.9	0.060

Closest-item-first reduction				
Strategy	UAS (%)	LAS (%)	LA (%)	Efficiency
SHIFT-LEFTA-RC-RIGHT-ARC	**70.75**	**68.95**	**91.0**	**0.079**
LEFT-ARC-SHIFT-RIGHT-ARC	66.48	64.74	90.7	0.058
SHIFT-RIGHT-ARC-LEFT-ARC	57.01	55.27	88.1	0.077
RIGHT-ARC-LEFT-ARC-SHIFT	52.55	51.11	87.8	0.056
LEFT-ARC-RIGHT-ARC-SHIFT	51.34	49.96	87.8	0.052
RIGHT-ARC-SHIFT-LEFT-ARC	51.34	49.96	87.8	0.051

9 Summary

Parse models are one of the main elements of data-driven parsers. They are used for guiding parsers during the processing of natural languages. However, it is possible that parse models may recommend no/several parse operations to a parser in a given situation. When parse models recommend no/several parse operations it is difficult for a parser to determine what operation to perform. Therefore, they are allowed to make deterministic choices. In this paper, we have identified several deterministic choices that a parser may take when it is presented with no/several parse operations, which are recommended by a parse model. We have observed and examined the effect of each deterministic choice on the performance of a data-driven parser, which is based on the shift-reduce algorithm. We have identified that each deterministic choice affects the parsing performance in different ways. Some choices affect accuracy while other choices affect efficiency.

Acknowledgment

Sardar Jaf's contribution to this work was supported by the Qatar National Research Fund (grant NPRP 09-046-6-001). Allan Ramsay's contribution was partially supported from the same grant.

References

Chang, C.-c. and Lin, C.-J. (2001), Libsvm: A Library for Support Vector Machines. *ACM Trans. Intell. Syst. Technol.* 3(2):1-27.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P. and Written, I. H., (2009), The weka data mining software: An update, *SIGKDD Explorations*. *Newsl*, 11(1):10-18.

Kuhlmann, M. and Nivre, J., (2010), Transition-Based Techniques for Non-Projective Dependency Parsing, *Northern European Journal of Language Technology,* 2(1):1-19.

Nivre, J. (2008), Algorithms for deterministic incremental dependency parsing, *Computational Linguistics,* 34(4): 513-553.

Attardi, G. (2006) Experiments with a Multilanguage Non-projective Dependency Parser, In Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X‘06, Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 166–170.

Maamouri, M. and Bies, A. (2004) Developing an Arabic treebank: methods, guidelines, procedures, and tools. In Proceedings of the Workshop on Computational Approaches to Arabic Script-based Languages. Geneva, pp. 2–9.

Nivre, J., Hall, J. and Nilsson, J. (2006), MaltParser: A Data-Driven Parser-generator for Dependency Parsing, In Proceedings of LREC.

Nivre, J., Rimell, L., McDonald, R. and Gomez-Rodriguez, C. (2010), Evaluation of dependency parsers on unbounded dependencies. In Proceedings of the 23rd International Conference on Computational Linguistics, COLING’10. Beijing, pp. 833–841.

Xia, F. and Palmer, M. (2001), Converting Dependency Structures to Phrase Structures, In Proceedings of the 1st Human Language Technology Conference (HLT 2001), San Diego. pp. 1–5.

Zhang, Y. and Clark, S. (2011), Shift-reduce CCG Parsing, In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1, (HLT 2011), Association for Computational Linguistics, Stroudsburg, PA, USA. pp. 683–692.

Prior Polarity Lexical Resources for the Italian Language

Simone Faro [1], Valeria Borzì [1], Arianna Pavone [2] and Sabrina Sansone [2]

[1] Dipartimento di Matematica e Informatica, Università di Catania,
Viale A.Doria n.6, 95125, Catania, Italy
{faro,borzì}@dmi.unict.it

[2] Dipartimento di Scienze Umanistiche, Università di Catania,
Piazza Dante n.2, 95100, Catania, Italy
{pavone,sansone}@unict.it

Abstract. In this paper we present SABRINA (Sentiment Analysis: a Broad Resource for Italian Natural language Applications) a manually annotated prior polarity lexical resource for Italian natural language applications in the field of opinion mining and sentiment induction. The resource consists in two different sets, an Italian dictionary of more than 277.000 words tagged with their prior polarity value, and a set of polarity modifiers, containing more than 200 words, which can be used in combination with non neutral terms of the dictionary in order to induce the sentiment of Italian compound terms. To the best of our knowledge this is the first prior polarity manually annotated resource which has been developed for the Italian natural language.

1 Introduction

The preparation of manuscripts which are to be reproduced by photo-offset requires special care. Papers submitted in a technically unsuitable form will be returned for retyping or cancelled if the proceedings cannot otherwise be finished on time. Sentiment classification, described in Bing and Lei (2012), Liu and Zhang (2012) and Medhat et al. (2014), concerns the use of automatic approaches for predicting the orientation of subjective content on text documents, with applications on many areas including information retrieval, customer intelligence and recommender and advertising systems.

Such discipline, where sentiment, opinion or emotion, are identified and classified in human written text is well known as sentiment analysis.

With the rapid increase of available subjective text on the internet in the form of blog posts, comments in discussion forums and product reviews, mining the user's opinion can assist in a lot of potential applications in areas such as recommender systems, search engines and market research.

Although some attempts have been made to extend solutions to other languages, till date all research efforts found in sentiment analysis literature deal mostly with English texts. However, in order to identify sentiment from a text, a lexical analysis of the source language plays a crucial role.

An approach for detecting sentiment in texts concerns the use of lexical resources such as a dictionaries of opinionated terms. For example the terms love , good and favorite directly indicate a positive sentiment or an opinion, while words like hate , bad and scandal can be associated with a negative sentiment.

Among the others, SentiWordNet, by Esuli and Sebastiani (2006), is one of the most used resource, containing opinion information on terms extracted from the WordNet database by Miller (1995) and made publicly available for research purposes. It is built via a semi supervised method and is considered a valuable resource for performing opinion mining tasks, providing a readily available database of term sentiment information for the English language.

Other previous works, as Pang and Lee (2002) and Esuli and Sebastiani (2006), have been already proposed for making dictionaries for those sentiment words using automatic approaches, however automatic identification of polarity orientation of such words is also a difficult research issue, known as polarity identification . In this context, it has been shown that the use of sentiment lexicons only provide a good baseline i.e. without using any natural language techniques only dictionary based approach produce a good performance, as noticed in Das and Bandyopadhyay (2010b).

An alternative to automatic tagged resources are manually annotated lexicons which turns out to be undoubtedly more trustable although they took long time to be constructed and may be subject it annotator bias.

In this paper we present SABRINA (Sentiment Analysis: a Broad Resource for Italian Natural language Applications) a manually annotated prior polarity lexical resource for Italian natural language applications in the field of opinion mining and sentiment induction. The resource consists in two different sets, an Italian dictionary of more than 277.000 words tagged with their prior polarity value, and a set of polarity modifiers, containing more than 200 words, which can be used in combination with non neutral terms of the dictionary in order to induce the sentiment of Italian compound terms. To the best of our knowledge this is the first prior polarity manually annotated resource which has been developed for the Italian natural language.

The paper is organized as follows. In Section 2 we introduce the concept of prior and posterior polarity and present some known lexicons which label terms with their sentiment polarity. Then in Section 3 we present the new tagged resources which has been created for the Italian language and discuss its properties. In Section 4 we briefly introduce also a web based fronted for accessing the resources. We draw our conclusions in Section 5.

2 Prior and Posterior Polarity

We would like to stress that the template should not be manipulated and that the guidelines regarding font sizes and format should be adhered to. This is to ensure that the end product is as homogeneous as possible.

A typical computational approach to sentiment analysis starts with prior polarity lexicons where entries are tagged with their prior out of context polarity as human beings perceive using cognitive knowledge.

The prior polarity of a term is the sentiment (positive or negative) that such word evokes by itself. More specifically we could define the prior polarity of a term as the polarity for its non-disambiguated meaning, out of any context.

For example the adjective cold evokes (in most cases) a fairly negative sentiment, since it is used in sentences as a cold man , a cold winter or I feel cold . However, depending on the context, we can find such term in sentences with a positive acceptation, as in I love cold beer.

In contrast with the prior polarity of a word, the polarities associated to each word sense is called in literature posterior polarity.

In most cases prior polarity lexicons are lists of positive and negative words, often deployed as baselines or as features for other methods for sentiment analysis research, as in Liu and Zhang (2012). In these lexicon, words are associated with their prior polarity. For example it is presumable that the term wonderful is associated with positive connotation while the term horrible is associated with negative one. These approaches have the advantage of not needing deep semantic analysis or word sense disambiguation to assign an affective score to a word and are domain independent. In other word they are less precise but more portable.

2.1 Polarity Lexicons

Opinion lexicons are resources that associate sentiment orientation and words. Their use in opinion mining research stems from the hypothesis that individual words can be considered as a unit of opinion information, and therefore may provide clues to document sentiment and subjectivity. These techniques could be broadly categorized in two genres: manual annotation and automatic extraction of word polarity.

Manual annotation. Manual annotated lexicons are undoubtedly trustable but it took long time and, for these reasons, tend to be constrained to a small number of terms. By its nature, building manual lists is a time consuming effort, and may be subject to annotator bias. Although such limitations manually created opinion lexicons were applied to sentiment classification as seen in Pang et al. (2002), where a prediction of document polarity is given by counting positive and negative terms.

Automatic detection. To overcome the above issues lexical induction approaches have been proposed in the literature with a view to extend the size of opinion lexicons from a core set of seed terms, either by exploring term relationships, or by evaluating similarities in document corpora. Early work in this area, by Hatzivassiloglou and McKeown (1997), extends a list of positive and negative adjectives by evaluating conjunctive statements in a document corpus. However in most cases automatic processes still demands manual validations and, moreover, may fail to cover the multiple domains as automatic processes trust on specific corpus.

SentiWordNet, by Esuli and Sebastiani (2006), is one of the most popular lexical resources in Sentiment Analysis. It has been widely adopted since it provides a broad-coverage lexicon, built in a semi-automatic manner, for English providing posterior polarities scores for each term of the language. It is the result of the automatic annotation of all the synsets of WordNet according to the notions of positivity, negativity, and neutrality. Different senses of the same term may thus have different opinion-related properties.

However in most opinion mining applications it is necessary to derive prior polarities starting from posterior polarities scores have been proposed in the literature. However, their performance varies significantly depending on the adopted variant. For instance SentiWords is an inducted prior polarity lexicon with the higher coverage for the English language. It contains roughly 155.000 words associated with a sentiment score included between -1 (strongly negative) and +1 (strongly positive), learned from SentiWordNet. Words in this resource are also aligned with WordNet lists. For the sake of completeness we notice also that other prior polarity sentiment lexicons are available for the English language, such as Subjectivity Word List, in Wilson et al. (2005), Word-Net Affect list, in Strapparava and Valitutti (2004), and the Taboada's adjective list, in Voll and Taboada (2007).

Although most of the efforts in literature have been devoted to the construction on lexicons resource for the English language, in recent years some research endeavors could be found in literature for solving the opinion mining problem in several languages and domains as in Das and Bandyopadhyay (2010b). Until date most of the approaches to sentiment analysis in languages different from English consists in applying a word-translation from the target language to English before polarity extraction, which is applied by using one of the above described lexicons. Such solutions, however, presents several problems including translation precision and disambiguation of words.

Recently some efforts have also been made to produce polarity lexicons for languages different from English. For instance Das and Bandyopadhyay (2010a) proposed multiple computational techniques like, WordNet based, dictionary based, corpus based or generative approaches for generating SentiWordNet for Indian languages.

For the sake of completeness we mention also an interactive gaming approach used for obtaining polarity values of english words, presented by Das and Bandyopadhyay (2010b) who proposed a tool, named Dr. Sentiment, to create and validate SentiWordNet in 56 languages by involving Internet population.

3 New Broad Lexical Resources for the Italian Language

In this section we present SABRINA[1] (Sentiment Analysis: a Broad Resource for Italian Natural language Applications) a manually annotated prior polarity lexical resource for Italian natural language applications in the field of opinion mining and

[1] A tool for evaluating SABRINA is available at the anonymous url http://www.dmi.unict.it/~faro/sabrina.

sentiment induction. The resource consists in two different sets, an Italian dictionary of more than 277.000 words tagged with their prior polarity value, and a set of polarity modifiers, containing more than 200 words, which can be used in combination with non neutral terms of the dictionary in order to induce the sentiment of Italian compound terms.

In recent years sentiment analysis in Italian texts has attracted attention due to Evalita, an initiative devoted to the evaluation of Natural Language Processing and Speech tools for Italian. In the recent Evalita 2014 edition the Sentipolc (SENTIment POLarity Classification) task[2] was proposed by Basile *et al.* (2014). It focused on Italian texts from Twitter by launching a battery of related tasks with an increasing level of complexity.

A first automatic annotated lexicon for the Italian language has been developed by Basile and Nissim (2013), who exploited three existing resources, namely MultiWordNet by Ciravegna et al. (1994), SentiWordNet by Esuli and Sebastiani (2006), and WordNet, by Miller (1995), to obtain an annotated lexicon of senses for Italian.

It was named Sentix and basically port the SentiWordNet annotation to the Italian portion of MultiWordNet in a completely automatic fashion. Sentix was then used by Castellucci *et al.* (2014) who described the UNITOR system that participated to the Sentipolc task within the context of Evalita 2014.

The system has been developed as a workflow of Support Vector Machine classifiers. Specific features and kernel functions have been used to tackle the different subtasks, i.e. Subjectivity Classification, Polarity Classification and the pilot task Irony Detection. To the best of our knowledge, besides Sentix, SABRINA is the first prior polarity manually annotated resource which has been developed for the Italian natural language.

Table 1. The distribution of polarity values assigned to Italian words

polarity	**value**	**# of words**	**% of words**
strongly negative	-1.0	22.651	8.17%
negative	-0.5	49.074	17.70%
neutral	+0.0	162.170	58.47%
positive	+0.5	36.688	13.23%
strongly positive	+1.0	6.739	2.43%

[2] http://www.di.unito.it/~tutreeb/sentipolc-evalita14/

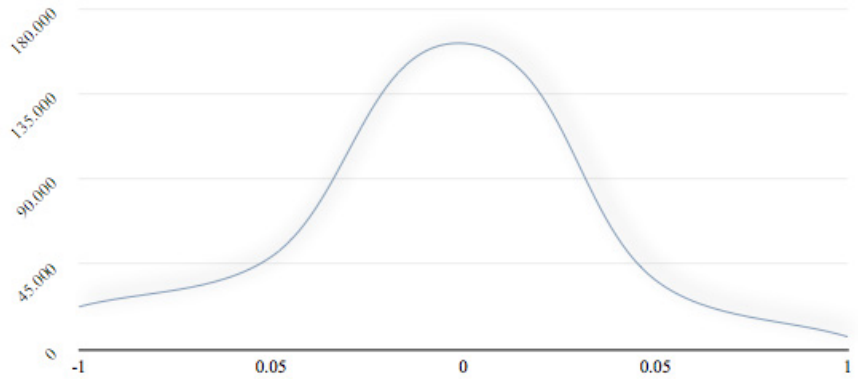


Fig. 1. The polarity distribution of the 277.387 different words of the Ispell Italian dictionary. Words are tagged with five different polarity values between -1 and +1

3.1 Italian Polarity Lexicon

Most sentiment lexicons in literature contain lists of tagged lemmas, i.e. the canonical form (or dictionary form) of a word. For instance the lastest version of MultiWordNet (1.39) contains around 58,000 Italian word senses and 41,500 lemmas organized into 32,700 synsets aligned whenever possible with Princeton WordNet English synsets. In using such kind of resources in sentiment analysis it is necessary to operate a previous step of sense disambiguation in order to identify the correspondent lemma of a word.

Our lexicon contains 277,387 words of the Italian language, including their inflection, used in order to express different grammatical categories such as tense, mood, person, gender, etc. For instance the dictionary contains the verb correre (to run) and its conjugations correvo , correrà , corressi , etc.

Such set of words have been manually tagged with their prior polarities. The annotation process started from the word set in the Ispell Italian dictionary[3] used for spell-checking purpose. Each word of the lexicon has been associated with a polarity in the range between -1 and +1 , where -1 indicates a strongly negative polarity while +1 indicates a very positive polarity. Mildly negative or positive opinion polarity have been tagged, respectively, with values -0.5 and 0.5 . In addition terms with a neutral polarity have been tagged with a value equal to 0 .

Two human annotators have been involved in the tagging process. The whole annotation process took more than three months.

Figure 1 shows the polarity distribution of all words of the Italian dictionary. We observed 162,000 words which have been tagged with a neutral sentiment polarity,

[3] Ispell is a program that helps you to correct spelling and typographical errors in a file. When presented with a word that is not in the dictionary, ispell attempts to find near misses that might include the word you meant.

more than 70,000 with a negative polarity and more than 43,000 words tagged with a positive polarity.

Specifically words evoking a negative sentiment are divided in two sets, 22,651 with a strongly negative polarity and 49,074 words with a fairly negative polarity. Similarly, in the case of words evoking a positive sentiment, we observed 6,739 words with a strongly positive polarity and 36,688 words with a fairly positive polarity. Table 1 shows in details the number of words detected for each polarity value together with the percentage of words detected in each group. Notice that more than 40% of words have been assigned to a polarity values, while 58% of words have been assigned with a neutral polarity.

3.2 Polarity Modifiers

An adjective is a word or set of words that modifies a noun or a pronoun. In most cases adjectives come before the word they modify. Some adjective can modify the polarity of a noun with a non neutral prior polarity. For example the adjective raro (rare) can be used in composition with the adjective bellezza (beauty) to emphasize its positive meaning (a women with a rare beauty). Similarly the adjective esiguo (scarse) can be used in combination with the noun valore (virtue) changing its positive polarity in a negative sentiment (a man with scarse virtue).

An adverb is a word or set of words that modifies verbs, adjectives, or other adverbs. Generally an adverb answers how, when, where, or to what extent an action is performed or an adjective is applicable. In this context some adverbs are able to modify the sentiment evoked by a verb or by an adjective with non neutral polarity. For instance the adverb appena (barely) can be associated with an adjective in order to reduce its positive (or negative) polarity, e.g. barely succeed or barely enthusiast . Similarly the adverb davvero (truly) can be associated with an adjective like sorprendente (amazing) in order to emphasize its positive meaning.

In our work we collected a set of more than 200 polarity modifier which have been manually tagged with a proportionality factor ranging between -2.0 and +2.0 . When a term with a non neutral polarity x is associated with a modifier with a proportionality factor y , we obtain a compound term whose polarity can be estimated as (x y) . Depending on the value of such factor we can distinguish four different kind of modifiers.

Emphasize. These modifiers have a proportionality factor greater than +1.0 and, when associated with a term having a non neutral polarity, evokes a sentiment which is stronger than the original one. thus they emphasize a positive (or negative) polarity value.

proprio bello (really beautiful) = +1.6 +1.0 = +1.6
alquanto sgradevole (rather unpleasant) = +1.5 -1.0 = -1.5
grande valore (great virtue) = +1.8 +0.5 = +0.9

Moderate. These modifiers have a proportionality factor greater than 0 and smaller than +1.0. When associated with a term having a non neutral polarity, they result in a compound term with a moderated sentiment which is weaker than the original one.

appena vinto (just gained) = +0.7 +0.5 = +0.35
mediamente brutto (ugly on average) = +0.5 -1.0 = -0.5
breve successo (brief success) = +0.6 +0.5 = +0.3

Reverse and moderate. This kind of modifiers have a proportionality factor greater than -1.0 and smaller than 0.0 . When they are associated with a term having a non neutral polarity, evoke a sentiment which is in opposition with the original sentiment, but has an absolute value of polarity which is smaller than the original polarity.

poco ragionevole (little reasonable) = -0.7 +0.5 = -0.35
esiguo dolore (scarse pain) = -0.7 -1.0 = +0.7
limitato guadagno (limited benefit) = -0.8 +1.0 = -0.8

Reverse and emphasize. These modifiers have a proportionality factor smaller or equal than -1.0 and, if associated with a term having a non neutral polarity, evokes a sentiment which is stronger than the original one but with an opposite polarity.

insufficiente prestigio (insufficient prestige) = -1.2 1.0 = -1.2
minime scomodità (minimal inconvenience) = -1.0 -0.5 = +0.5
scarso valore (lacking virtue) = -1.2 +0.5 = -0.6

4 A Web Based Frontend

We implemented a simple web based tool in order to access the lexical resource presented in this paper. In order to allow a blind review of the paper we uploaded the tool in a free hosting server. The tool is accessible at the url

http://www.dmi.unict.it/~faro/sabrina

The tool allows to evaluate single Italian terms or compound terms, where words with a non neutral polarity are associated with modifiers, as described above. Moreover each example which you can find above in the paper is tagged with an anchor which redirect the reader to the web page of the tool in order to evaluate the sentiment value of the example itself.

If a whole sentence is tested by the tool, containing more than one term with non neutral prior polarity, then a straightforward approach is applied in order to compute an approximation of the polarity of the whole sentence. In particular the set of polarity values contained in the sentence is arranged from the lowest one to the highest one and the median of such a set is taken as the polarity value of the whole sentence. Specifically the median is the number separating the higher half of the set of polarity values from the lower half. If there is an even number of polarity values, then there is no single middle value. Int this cases the median is usually defined to be the mean of the two middle values.

5 Conclusions

In this paper we presented a new lexical resource for the Italian language containing more than 277.000 words which have been manually tagged with their prior polarity values, i.e. a value indicating the sentiment which such words evoke when are out of any context. We also provide an additional lexical resource containing a set of more than 200 polarity modifiers which can be used for inducing the sentiment polarity of Italian compound terms. Future works will be devoted to test the effectiveness of such resource in opinion mining task.

References

Basile V. and Nissim M. (2013) Sentiment analysis on Italian tweets. In Proceedings of the 4th Ws: Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 100–107.

Basile V., Bolioli A., Nissim M., Patti V., and Rosso P. (2014) Overview of the Evalita 2014 SENTIment POLarity Classification Task. In Proceedings of the 4th evaluation campaign of NLP and Speech tools for Italian (EVALITA), Pisa, Italy.

Castellucci G., Croce D., De Cao D., Basili R. (2014) A Multiple Kernel Approach for Twitter Sentiment Analysis in Italian. In Proceedings of the First Italian Conference on Computational Linguistics (CLIC-IT).

Ciravegna F., Magnini B., Pianta E., Strapparava C. (1994) A Project for the Construction of an Italian Lexical Knowledge Base in the Framework of WordNet IRST Technical Report #9406-15.

Das A. and Bandyopadhyay S. (2010) SentiWordNet for Indian Languages. Proceedings 23rd International Conference on Computational Linguistics, pages 56–63.

Das A., Bandyopadhyay, S. (2010) Towards the Global SentiWordNet. Proceedings of the 24th Pacific Asia Conference on Language, Information and Computation.

Bing L. and Lei Z. (2012) A Survey of Opinion Mining and Sentiment Analysis. Book Chapter, Mining Text Data, Springer US, pages 415–463.

Esuli A. and Sebastiani F. (2006) SentiWordNet: A publicly available lexical resource for opinion mining. In Proceedings ofLREC.

Hatzivassiloglou, V., and McKeown, K. (1997). Predicting the Semantic Orientation of Adjectives. Preceedings of the 35th Annual Meeting of the Association of Computational Linguistics (ACL'97). Madrid, Spain, pp. 174-181.

Liu B. and Zhang. L. (2012) A survey of opinion mining and sentiment analysis. Mining Text Data, pages 415–463.

Medhat W., Hassan A., Korashy H., (2014) Sentiment analysis algorithms and applications: A survey. Ain Shams Engineering Journal, Volume 5, Issue 4, Pages 1093–1113.

Miller G. A. (1995) WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.

Pang B., Lee L., and Vaithyanathan, S. (2002) Thumbs up? Sentiment Classification using Machine Learning Techniques. Proceedings of EMNLP.

Strapparava C. and Valitutti A. (2004) WordNet-Affect: an affective extension of WordNet. In Proceedings of LREC 2004, pages 1083–1086, Lisbon.

Voll K. and Taboada M. (2007) Not All Words are Created Equal: Extracting Semantic Orientation as a Function of Adjective Relevance. In Proceedings of the 20th Australian Joint Conference on Artificial Intelligence. pages. 337-346.

Wilson T., Wiebe J. and Hoffmann P. (2005) Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. Proceedings of HLT/EMNLP 2005.

An Orthography Transformation Experiment with Czech-Polish and Bulgarian-Russian Parallel Word Sets

Andrea Fischer, Klara Jagrova, Irina Stenger,
Tania Avgustinova, Dietrich Klakow and Roland Marti

Saarland University, Collaborative Research Center (SFB) 1102:
Information Density and Linguistic Encoding,
Campus A 2.2, 66123 Saarbrücken, Germany
Project C4: INCOMSLAV
Mutual Intelligibility and Surprisal in Slavic Intercomprehension
{kjagrova, avgustinova}@coli.uni-saarland.de,
{ira.stenger, rwmslav}@mx.uni-saarland.de,
{andrea.fischer, dietrich.klakow}@lsv.uni-saarland.de
http://www.sfb1102.uni-saarland.de

Abstract. This article presents the methods and findings of a computational transformation of orthography within two Slavic language pairs (Czech-Polish and Bulgarian-Russian) on different word sets. The experiment aimed at investigating to what extent these closely related languages are mutually intelligible, concentrating on their orthographies as linguistic interfaces to the written text. Besides analyzing orthographic similarity, the aim was to gain insights into the applicability of rules based on traditional linguistic assumptions for the purposes of language modelling.

1 Introduction

We are interested in identifying the mechanisms by which languages en- and decode information, focusing on the phenomenon of receptive multilingualism observed within the Slavic language group. We are framing the problem as one of (statistical) language model adaptation from a L1 to L2, incorporating results from traditional approaches and comparative historical linguistics. The key idea is that comprehension of a text in an unknown, but related language should be better when the language model adapted for processing the unknown language exhibits relatively low average surprisal.

This contribution elaborates on an inter-language orthographic transformation experiment[1] for which, based on orthographic features, different mappings between selected language pairs were tested. Two language pairs for which a relatively high degree of mutual intelligibility could be expected were chosen: Czech-Polish (CS-PL,

[1] The experiment took place in the initial phase of the INCOMSLAV project at Saarland University, launched in October 2014. Morphology, lexis and syntax will be subject to later project phases.

both West Slavic, and both using the Latin script with a number of additional diacritic signs) and Bulgarian-Russian (BG-RU, South and East Slavic, both using Cyrillic script). The probably best known and most obvious example for such orthographic correspondences of characters between Czech and Polish are *v:w*, *h:g*, *č:cz*, etc.

We collected and systematized traditional linguistic assumptions about how Slavic languages developed from a reconstructed parent language – referred to as Proto-Slavic or Common Slavic – to the modern varieties of Czech, Polish, Bulgarian and Russian (Schenker 1993). Although this parent language existed before any Slavic script appeared, historical linguistics was able to reconstruct how and in which stages the individual modern varieties moved from unity to diversity in the course of several centuries (Carlton 1991:9). The key features which reflect this development and now distinguish one Slavic language from another had their origin in Proto-Slavic times. Thus, there is a common base in the linguistic systems of the individual languages.

The existing orthography rules can be considered a result of both linguistic and sociolinguistic factors (Sgall 1987; Penzl 1987). Orthography does not only follow phonological, morphological and diacritical principles. It is also the syntactic, semantic, etymological and historical factors that are reflected in the graphematic representation of a language. Apart from this, written language is subject to manipulation by rules and laws created by governing authorities (e.g. in the process of spelling reforms). Kučera explains the specific character of Cyrillic as follows:

> *"Like Glagolitic and unlike the Latin alphabet, Cyrillic was a script customized to the contemporaneous Slavic languages, with a highly efficient and systematic one-to-one correspondence between its graphemes and the Slavic set of phonemes. […] [T]here have been few exceptions from the correspondence, a fact that was in marked contrast with the widespread use of digraphs in the systems based on the Latin alphabet. Thus, there was significantly more asymmetry, and consequently more looseness in the relation of the Slavic phonemes to the Latin graphemes than in their relation to Cyrillic graphemes." (Kučera 2009:74)*

2 Experimental Setup

Parallel contemporary vocabulary lists were analysed in terms of their orthographic similarity and the applicability of the correspondence patterns that are assumed in comparative Slavic linguistics. The objective of our transformation experiment was, in the first place, to validate (confirm or reject) the traditional assumptions by applying orthographic correlation rules, which were formulated on the basis of historical comparative linguistics, on contemporary word material. As a result of this experiment it should be possible, with the help of the validated rules, to describe or even predict the written representation of units of the source language a target language. If however the traditional assumptions appear not to hold for certain vocabulary (sub-) sets, the relevant orthographic correlations were to be directly derived from the compiled parallel word lists.

2.1 Rules Inferred from Traditional Linguistic Assumptions

To account for the historically conditioned variation between the languages under investigation, we first collected and worked out orthographic correlations reflecting the development of the sound systems as established in historical comparative linguistics. We attempted to accommodate the main lines of the sound system evolution, from Common Slavic to individual modern Slavic languages, focusing on the following aspects: (i) development of vowels and consonants, (ii) development of specific sound combinations, and (iii) the metathesis of liquids.

The next step when designing the rule sets for the transformation experiment was a change of perspective, away from the perspective *Common Slavic vs. all other* towards a comparison of language pairs. In the diachronically-based language-family-oriented collection of correspondences[2] there were 132 for CS-PL vs. 126 for BG-RU (i.e. *h:g:ɔ:ɔ* for CS-PL-BG-RU). A considerable number of these rules stated regular one-to-one correspondences for the respective language pairs, for example such rules as *p:p* for CS-PL and *к:к* for BG-RU. Consequently, only those rules were applied in the experiment that represent a mismatch between target and source language units (e.g. *č:cz* for CS-PL and *ъ:y* for BG-RU), so that only 81 rules for CS-PL and 48 rules for BG-RU were applied to the word lists. This suggests a greater orthographic diversity between Czech and Polish than between the other two languages. Equal-to-equal grapheme correspondences were not considered a transformation. Such a situation in fact represents a reading intercomprehension scenario in which equal graphemes are not expected to cause any additional surprisal for readers. The remaining transformation rules were then applied on parallel word lists and checked for their practical usability.

Czech and Polish: Although both use the Latin script, they differ in their diacritical systems and the use of digraphs. While CS sibilants are usually represented by a single character, PL uses digraphs instead, at least for hard sibilants, e.g. *č:cz*. In the experimental setup, a letter is defined as an independent unit including diacritics, if applicable. For the purposes of the current automatic transformation, digraphs are considered two characters, e.g., PL *sz* and CS *ch*. There are 15 Czech letters (*á, č, ď, é, ě, í, ň, ř, š, ť, ú, ů, v*[3], *ý, ž)* that do not exist in PL, and 9 Polish letters (*ą, ć, ę, ł, ń, ś, w*[4]*, ż*, *ź*) that do not exist in CS. Still, these letters are expected to be legible for readers of the respective target language (i.e. by ignoring diacritical signs) and thus should not impair reading intercomprehension to a large extent – especially when the actual phonetic representation is similar (e.g. *á* vs. *a*, although this fact might not be known to the reader).

Bulgarian and Russian both use the Cyrillic script and there are only slight differences in the alphabets. The use of digraphs and diacritics is rare in the Cyrillic-based systems. The Russian letters э, *ё*[5], *ы* do not appear in BG. Generally, one can

[2] The analyses were primarily collected from Bidwell (1963), Žuravlev et al. (1974-2012) and Vasmer (1973).

[3] The letter *v* is only used in Polish texts when it is part of a named entity or a foreign word.

[4] The letter *w* is only used in Czech texts when it is part of a named entity or a foreign word.

[5] The letter *ë* is used mostly only in dictionaries and schoolbooks.

distinguish two important orthographic differences: unfamiliar graphemes representing unfamiliar or familiar phonemes (these differences only apply to a limited number of graphemes between BG and RU); graphemes that seem to be familiar, but in fact the grapheme-phoneme correspondences are different (e.g. ъ and щ in BG are pronounced [ə] and [ʃt], while their RU counterpart ъ has no phonetic, but an orthographic function (hard sign) and щ is pronounced [ʃʧ], different rules for the reduction of unstressed vowels etc.).

2.2 Word Sets Used

In the initial phase of INCOMSLAV, we started collecting all parallel word lists and corpora that were available to us in digital format. The main inspiration and the first source of Slavic word lists was the EuroComSlav website. We decided to test the traditional assumptions on word lists instead of full texts in order to focus on the orthographic level only and thus exclude such influences that are caused by individual morphological rules from our analysis as far as possible.

Verb forms play a special role in the BG-RU comparison. While we analyzed infinitive verb forms in the CS-PL lists, we had to replace all infinitives in the BG-RU lists with the 3rd person present tense forms of the verbs. This was done to ensure a more appropriate comparison of RU with BG, as there are no infinitive forms in BG and 1st person forms are highly irregular, which makes them less suitable for an orthographic comparison.

There were three types of basic parallel lists available for all four languages: a Pan-Slavic list and a list of internationalisms on the EuroComSlav website, and the online version of the Swadesh list. The EuroComSlav lists had to be corrected for errors. All lists were slightly modified, as formal non-cognates (i.e. CS-PL *mnoho – wiele* [*many*/*much*]; BG-RU *ние – мы* [*we*]) were removed and formal cognates, if existing, were added to the lists, where the pairs consisted of non-cognates (i.e. *mężczyzna* [*man*] substituted by *mąż* [*husband*] in CS-PL *muž – mąż*; *звяр* [*beast*] added to its RU formal cognate *зверь* [*animal*, *beast*] for the BG-RU pair *зверь – звяр*). Focusing only on the formal aspect of the lexemes, we did not take semantics into account. This explains the variation in the amount of words for each list in each language pair.

Table 1. Word sets with numbers of items

	Total number of items	
Word list	CS-PL	BG-RU
Swadesh list	212	227
Panslavic list	455	447
Internationalism list	262	261
Homonyms	1553	X
Dictionary	80963	X

For the CS-PL pair we implemented two additional large word lists which might have a statistically more representative effect: A set of homonyms, extracted from (Szałek and Nečas 1993), as well as an open-source digital version of a CS-PL dictionary containing more than 80,0000 lexemes (Kazojć 2010).

2.3 Method

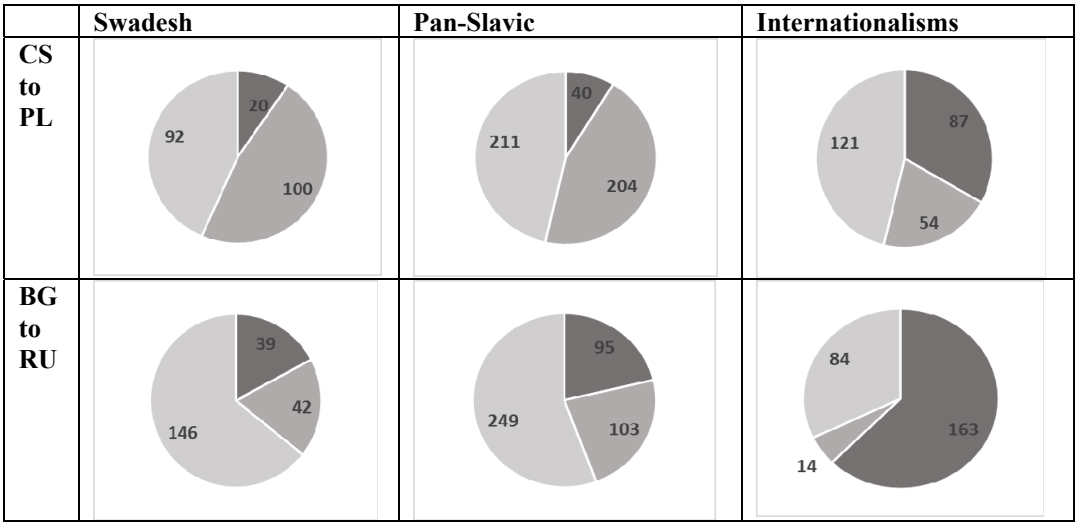
If all characters in a word of L1 are the same as in the corresponding word in L2, the word was automatically listed as *input identical*. If there is a mismatch of one or more positions in the word pair, the computer tries to apply one or more rules from the transformation rule set. If all characters in a word of L1 can be transformed with the help of the rules into the L2 word, the word pair is listed as *correctly transformed*. Rules for strings of characters take precedence over rules for single characters. There is also a chance that a unit from L1 corresponds to a different unit (character or string of characters) in L2, which is not part of the traditional linguistic rule set entered for this experiment. In such a case, these words are classified as *untransformed*.

The computer code for the implementation of the orthographic transformation rules between language pairs (by Andrea Fischer and Ali Shah) is provided below.

```
method Transformations(w, T)
--------------
input: a word w from language L1, the set T of admissible transformation rules
output: all L2 transformations of w obtained by applying rules from T
--------------

transformations = {(w, [])} // initialize the set of transformations with just
the word and no applied transformations

new_variants = {} // temporary iteration variable

while True: // iterate until no new transformations are found anymore
        for t in T: // process each transformation rule
                for variant, path in transformations: // apply this rule to
                all currently known variants of the original word
                        for new_word, application_pattern in Transform-
                        WithRule(variant, t): // apply the rule t in each com-
                        bination of positions where it is applicable
                                new_variants.add((new_word, path, applica-
                                tion_pattern)) // record the new variant plus
                                the path by which it was obtained
        if words(new_variants) + words(transformations) ==
        words(transformations): // after processing all rules, see if there are
        any new words
                break // stop iteration if no new words were found
        else:
                transformations.addAll(new_variants) // record the newfound
                transformations and continue iterating otherwise

return transformations
```

2.6 Results of the Implementation of the Rules

Fig. 1. Results of implementation for both language pairs



Legend: input identical, correctly transformed, untransformed

The most obvious finding is the different proportion of orthographically identical words in the language pairs (max.: 33.21 % for CS-PL vs. 62.45 % for BG-RU). The internationalism lists, consisting only of nouns, show the highest proportion of orthographically identical words in both pairs. An explanation for the low rate of identical words in the BG-RU Swadesh list is the high rate of morphological differences reflected in orthography, e.g. different endings of male adjectives and verb forms in 3rd person singular – here the orthographic rule set can be applied only in very few cases. However, the rule set works well for the CS-PL Swadesh list (best transformation rate of the experiment: 47.17 %).

The Swadesh lists consist of a relatively high rate of verbs and adjectives and they are the only lists containing a number of pronouns, prepositions and numerals. The Pan-Slavic lists include nouns, verbs and adjectives. While for CS-PL the proportion of untransformed items is relatively constant throughout the three lists, the untransformed part for BG-RU ranges from 64.32 % with the Swadesh list to 32.18 % with internationalisms.

Tables 2a and 2b display the five most frequently used rules for each word set, given the rule from L1 to L2 along with the number showing how often this rule was applied in words that were classified as *correctly transformed*. Directly under the rules, one example word pair for which this rule holds is provided.

Table 2a. Most frequent transformation rules applied on the different lists

Czech to Polish

Swadesh	Pan-Slavic	Internat.	Homonyms	Dictionary
t:ć,24	ý:y,45	á:a,15	v:w,307	v:w,991
dát–dać	dýně–dynia	bál–bal	věc–wiec	kráva–krowa
ý:y,21	v:w,42	e:a,12	ý:y,175	t:ć,663
nový–nowy	voda–woda	linie–linia	výlet–wylot	prát–prać
v:w,20	t:ć,37	v:w,8	t:ć,163	á:a,515
dva–dwa	bolet–boleć	káva–kawa	tma–ćma	pára–para
á:a,10	l:ł,24	í:e,5	á:a,142	e:a,353
já–ja	zlý–zły	talíř–talerz	čára–czara	duše–dusza
l:ł,9	h:g,20	l:ł,4	l:ł,111	ý:y,336
teplý–ciepły	hlava–głowa	kanál–kanał	látka–łatka	dým–dym
		rá:ra,4		
		rádio–radio		

CS-PL: The success of *t:ć* can be explained by the high rate of verb endings (morphological feature reflected in orthography) in all lists except in internationalisms, although this rule was originally inferred from the diachronically-based rule for *deset - dziesięć*. Another outstanding rule is *ý:y* which is due to a high rate of adjective endings in the lists, although this rule was originally derived from a historical correspondence in word stems. For some rules such as *á:a* and *l:ł* it may be assumed that they will not pose a problem to reading intercomprehension because the diacritics can be ignored. The *v:w* rule represents characters that would not appear in the other language. The success of applying these rules in this experiment depends strongly on their overall frequency of the individual characters in the word lists, i.e. there is a higher frequency of *h:g* in Pan-Slavic vocabulary relative to *h:g* in all other lists. The strongest tendencies for vowel changes are from *e:a*, from *í:e*, which both apply for noun endings. As a result, the findings reveal a strong applicability of rules that refer to endings, to letters that are not part of the inventory of one language and to letters that are only distinguished by the absence or presence of diacritical signs.

Table 2b. Most frequent transformation rules applied on the different lists

Bulgarian to Russian

Swadesh	Pan-Slavic	Internation.
ъ:у,8	я:е,17	л:ль,9
път–путь	вяра–вера	цел–цель
я:е,7	ъ:у,10	и:ы,1
цвят–цвет	дъб–дуб	музика–музыка
и:ы,6	и:ы,10	н:нь,1
ти–ты	диня–дыня	амин–аминь
е:я,6	е:я,9	р:рь,1
език–язык	ред–ряд	календар–календарь
е:о,4	е:ё,8	а:я,1
езеро–озеро	еж–ёж	плаж–пляж

BG-RU: The most frequent orthographic correspondences of the transformation experiment in the Swadesh and Pan-Slavic lists are between the orthographic representations of vowels: *ъ:у; е:я; я:е; и:ы; е:о; е:ё*. The orthographic differences could generally be explained, on the one hand, by the different development of the vowels from Common Slavic to the modern Slavic languages and, on the other hand, by subsequent spelling reforms in these languages with the aim to harmonize their writing system to the sound system, i.e.:

ъ:у – explained by the different development of the back nasal vowel */ǫ/ of Common Slavic to /□/ in Bulgarian and to /u/ in Russian.

е:я – also explained by the different development of the front nasal vowel */ę/ of Common Slavic to /e/ in Bulgarian and to /ʹa/ in Russian.

The most frequent orthographic correspondences in the internationalism list, besides the correlations of orthographic representations of the vowels *а:я* and *и:ы*, here concern the orthographic representation of consonants, e.g. *л:ль*, *н:нь*, *р:рь*, which can be explained by the difference between non-palatalized consonants in Bulgarian and palatalized consonants in Russian. It must be kept in mind, however, that most internationalisms in the list are borrowings from other languages and thus constitute a rather specific problem. Usually, in orthographies using Cyrillic the pronunciation of the borrowing may be preserved and the spelling may be changed to correspond to the orthographic rules of the borrowing language (Kučera 2009). Borrowings were generally handled in harmony with the phonological and morphological principles of each particular language, which could be presented by other orthographic correspondences that are distinct from our diachronically-based transformation rules. This could be an explanation for the fact that only five of the transformation rules could be successfully applied on the internationalism list. However, there already is a high rate of identical words in this list.

The overall results for both language pairs show that there are different principles in how the diachronically-based transformation rules work. For BG-RU, the re-

sults confirm the validity of the rule set to a high degree for the reasons mentioned above. For CS-PL we found that the traditional rules were valid not only for word stems as explained in historical comparative research, but also for other parts of words, mainly endings. The rules do not only cover orthographic features, but also those morphological features to which the same rules apply. The words classified as *correctly transformed* were much lower in number for BG-RU. This could be explained by the fact that in the experiment, words in which there was only one unit that could not be transformed with the rule set, were sorted out by the program as *untransformed*. For example the adjective pair *mux* (BG) vs. *muxuй* (RU) could not be correctly transformed, because there is no rule saying ø [nothing] in BG corresponds to -*uй* in RU – this would require a morphological rule set.

The difference in the language pairs confirms the isolated position of Bulgarian in contrast to the other languages under focus, especially because of its morphology.

3 Conclusions

In the present application of diachronically-based orthographic transformation rules between the two language pairs CS-PL and BG-RU we tried to find out to what extent traditional linguistic assumptions explain the differences between parallel word sets in the languages. The computational transformation experiment revealed that there are different percentages of orthographically identical words in both language pairs. For all word sets, the initial orthographic similarity is greater for BG and RU (max.: 62.45 % for internationalisms) than for CS and PL (max.: 33.21 % for internationalisms), which suggests a greater degree of mutual intelligibility for BG-RU by the presence of internationalisms than in the other pair.

For those words in the parallel lists that were not identical in terms of orthography, a rule set of inter-language orthographic correspondences was applied. For the CS-PL combination, these orthographic transformation rules led to better results – 44.84 % for the Pan-Slavic vocabulary list, while the results for BG-RU in the same list amounted to only 23.04 %. The low success rate for the BG-RU orthographic transformations suggests a high influence of morphological differences between these languages (zero endings for BG adjectives, different verb endings, etc.). While investigating the CS-PL orthographic correspondences, we found that the morphological features are reflected in the respective orthographies to a similar degree and are therefore comparable. This suggests that knowledge of those orthographic correspondence rules might improve reading comprehension, e.g., for a Czech native speaker reading Polish. The knowledge of orthographic correspondences between BG and RU, in contrast, is not expected to lead to such large improvement in reading comprehension as in the other pair, when the respective other language is unknown to the reader. However, knowledge of morphological cross-language correspondence principles might be much more helpful here.

4 Outlook

Orthography was subject to the first of six work packages in the INCOMSLAV project. In the near future a series of on-line reading (inter-)comprehension experiments with Slavic native speakers is planned to validate the findings of this and other computational experiments. The results from the experiments with human readers will be discussed in the framework of several other computational estimations and calculations of similarity and distance. The upcoming project phases will cover morphology, lexis and syntax. On the linguistic level, more similarities and discrepancies in the subsystems of the languages will be investigated. Both the nature of the phenomena and the strength of the effects are relevant at these levels.

For the information-theoretic part of the project, the aim will be to adapt feature-based n-gram language models for cross-language use via latent space and similarity. The information-theoretical results will then be analyzed again from a linguistic point of view and interpreted together with the results of reading intercomprehension experiments with Slavic native speakers.

References

Bidwell, C.E. (ed) *Slavic Historical Phonology in Tabular Form.* Mouton & Co., The Hague, 1963

Carlton, T. R. (ed) *Introduction to the Phonological History of the Slavic Languages.* Slavica Publishers, INC. Columbus, Ohio, 1991

Kučera, K. (2009) The Orthographic Principles in the Slavic Languages: Phonetic/Phonological. In: Kempgen, S., Kosta, P., Berger, T., Gutschmidt, K. (eds.) *The Slavic Languages. An International Handbook of their Structure, their History and their Investigation*. Volume 1. Walter de Gruyter, Berlin/New York, pp. 70-76

Penzl, H. (1987) Zur alphabetischen Orthographie als Gegenstand der Sprachwissenschaft. In: Luelsdorff, P. A. (ed.): *Orthography and Phonology*. John Benjamins Publishing Company, Amsterdam/Philadelphia, pp. 225-238

Schenker, A.M. (1993) Proto-Slavonic. In: Comrie, B., Corbett, G.G. (eds.) *The*

Slavonic Languages, Routledge, London and New York, pp. 60-125

Sgall, P. (1987) Towards a Theory of Phonemic Orthography. In: Luelsdorff, P. A. (ed.) *Orthography and Phonology*. John Benjamins Publishing Company, Amsterdam/Philadelphia, pp. 1-31

Dictionaries

Szałek, M.; Nečas, J. (eds) *Czesko-Polska Homonymia*. Poznań, 1993

Vasmer, M (ed) *Etimologičeskij slovar' russkogo jazyka*. Moscow, 1973

Žuravlev, A. F., et al. *Etimologičeskij slovar' slavjanskich jazykov*. Vyp. 1-37. Moscow, 1974-2012

Online documents

Swadesh list:
http://en.wiktionary.org/wiki/Appendix:Swadesh_lists_for_Slavic_languages.
Accessed 22/04/2015

Pan-Slavic list:
http://www.eurocomslav.de/kurs/pwslav.htm. Accessed 22/04/2015

Internationalism list:
http://www.eurocomslav.de/kurs/iwslav.htm. Accessed 22/04/2015

Kazojć, J. (2010) Otwarty słownik czesko-polski V.03.2010 (c)
http://www.slowniki.org.pl/czesko-polski.pdf. Accessed 22/04/2015

Hierarchies of Terms on the Euromaidan Events: Networks and Respondents' Perception

D. Lande, A. Snarskii, E. Yagunova, E. Pronoza and S. Volskaya

Institute for Information Recording NAS of Ukraine, Kiev, Ukraine
NTUU "Kiev Polytechnic Institute", Kiev, Ukraine
{dwlande, asnarskii}@gmail.com
Saint-Petersburg State University, Saint-Petersburg, Russian Federation
{iagounova.elena, katpronoza, svetlana.volskaya}@gmail.com

Abstract. In this paper we describe the construction methodology of a network of natural terms hierarchy on the base of the subject arrays of news texts. The proposed method is illustrated using automatic processing of the full texts of the articles about the Euromaidan events in Kiev.

1 Introduction

Constructing a large domain-specific ontology is a challenging problem. The ontology development process includes such a task as terms learning, but the problem of effective unsupervised terms learning is unsolved, and the problem of the links identification and automatic network construction is also still open.

Another important task is the formal estimation of the number of new topics in data streams. Appearance of new topics naturally causes appearance of the series of terms marking new themes. A linguist dealing with news texts has to know the specifics of different segments of media data streams. Particularly, sometimes one can correlate separate news topics with the subjects of whole data flows using lexical features.

In this paper we propose an approach to the construction of a terminological basis for interrelated events, which are described in the messages of electronic media, and for separate subjects of data flows for a certain time period. We also consider some principles of making a language network on the base of the selected terms. Correlation of unit message terminology with general subject terminology can be considered as a formal criterion of event relevance to the considered subject area (sequence of events).

The problems of events modeling and analyzing their perception by the informants have been an object of many recent studies [0]. Unsupervised terms extraction task is also widely addressed by the researchers. Terms extraction methods are either statistics-based (e.g., clustering [0]), or use fine-grained linguistic analysis (e.g., dependency parsing [0]). Some researchers also employ external sources of knowledge like Wikipedia or Wordnet [0]. Our method is statistics-based. It is fast and language independent and does not demand any linguistic resources.

2 Data

The data for our research consists of news reports about the confrontation in Kiev in 2013-2014, which was caused by so-called Euromaidan. We collected more than 200 thousand of news reports from RuNet web sites during the period from November 2013 till March 2014.

First of all, it is necessary to choose a text corpus for the further analysis. To collect the data for our research, we use "InfoStream" – a system of content monitoring. To retrieve the news reports which are relevant to the subject area we make the following request:

(maydan|euromaidan)&(beat|dispersal|storm|
berkut|molotov|titushk|was killed) & lang.RUS.

The collected corpus consists of more than 200 thousand of news reports. On the base of the corpus the dynamics of subject reports should be identified. The mode «Dynamics of events» in the system of content monitoring «InfoStream» allows getting information about the number of published articles which are relevant to the request for a certain time period. This information is presented in the form of a plot (see Fig. 1).

Fig. 1. Dynamics of the number of publications which are relevant to the request

The time dynamics data is normalized for each day, and time series is built. Each relative frequency value in this series equals the ratio of the number of subject reports per day to the number of all the reports per day. It allows us to ignore weekly periodicity in the number of subject reports.

After we get the information about publications dynamics, the critical points should be identified. These critical points are the local maxima of time series in the dynamics of publications [3].

On the base of these results three dates were chosen (2013.11.30, 2014.01.22, 2014.02.19) as critical points for the sequence of events under consideration.

After the critical points are selected, it is necessary to extract the main sequences of subject reports which are relevant to the request for the necessary dates (see Fig. 2). It is also done via the system of content monitoring.

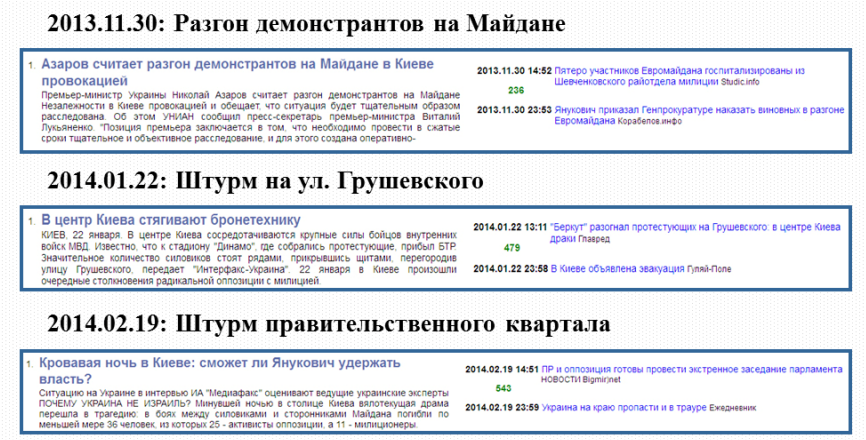


Fig. 2. Main subject concatenations for necessary dates

3 Construction method of Network of Natural Term Hierarchy

3.1 Extraction of Terms for Ontology

For the further analysis we build three corpora from the reports. Each corpus corresponds to one of the three found critical points; lexical features of each corpus are the objects of monitoring.

Preprocessing of these corpora includes division of text into fragments (separate reports, paragraphs, sentences, words, bigrams, and trigrams), deletion of analphabetical symbols and cutting off inflections – stemming (this is an option).

Then each term from the text (unigram, bigram or trigram) receives an estimation of its «discriminant power», represented by TF-IDF. The preliminary technique description was published in [12].

3.2 Construction of Terms Hierarchy

The process of network constructing is based on using semantically important text elements. To identify these elements in the text one can use methods described in [0], [0] and [0]. An advantage of a network built on the base of important text elements, pivot words and words combinations is that such a network embraces separate knowledge domains.

Extracting of the terms for a network is done using the feature based on the discriminant power of words. Nevertheless one should remember that this feature cannot guarantee high quality of ontology. Most frequent words from the chosen subject area, which have low discriminant power (for example, the words “Ukraine”, “Mai-

dan", "Protest" in the news corpus about Euromaidan in Kiev) could be the most important ones for the network construction.

The content of the corpora is the base of the future network. In this work we consider a natural network. We call the network natural due to the fact that its construction does not include any special methods of semantic analysis (including part of speech tagging). All the relations in this network are determined by the positions of the words and word combinations, which are extracted from the texts of statistically significant size. Terms hierarchy which is built completely automatically is the base for the further automatic ontology construction with experts.

In our work we propose a method of constructing terms hierarchy which includes the construction of a compactified horizontal visibility graph (CHVG) and terms weights recalculation (for unigrams, bigrams and trigrams) [0].

Language network is built in three stages using the CHVG algorithm.

1. In the first stage nodes sequence is marked on the horizontal axis. Each node corresponds to the word in the order it appears in the text. On the vertical axis TF-IDF weights are put. Vertical lines are drawn between these TF-IDF values and their projections on the x-axis.

2. In the second stage a traditional horizontal visibility graph is built [0]. An edge is drawn between every two nodes if these nodes are in "direct visibility". "Direct visibility" of the nodes means that they can be connected by a horizontal line which does not intersect any vertical line in the plot.

3. In the third stage we merge the nodes with the same words. The edges of such nodes are also merged. Such procedure is called graph compactification. Node weights are recalculated. TF-IDF values are replaced with the corresponding node degrees in CHVG. Finally, the terms are sorted according to their new CHVG weights in descending order. Stop words are excluded from further analysis. In this paper a list of stop words is formed using following web-resources: https://code.google.com/p/stop-words/downloads/list, http://www.ranks.nl/stopwords/, http://www.textfixer.com/resources/common-english-words.txt.

Experts estimate the size of the network (let us denote it by N). Then N unigrams, N bigrams and N trigrams with the largest CHVG weights are selected. The network is constructed using the obtained terms. In this network nodes identify terms and links represent part-whole relations between the terms. Fig. 3 presents an example of the terms hierarchy construction. Different geometric figures denote different words in Fig. 3. Unigrams are grouped in the first column, while bigrams and trigrams are in the second and third columns respectively. If a unigram belongs to some bigram, or a bigram is a part of some trigram, an arrow is drawn between them (denoting a part-whole relation). The set of terms together with the links between them forms a three-level Natural Network of Terms Hierarchy [0], [12].

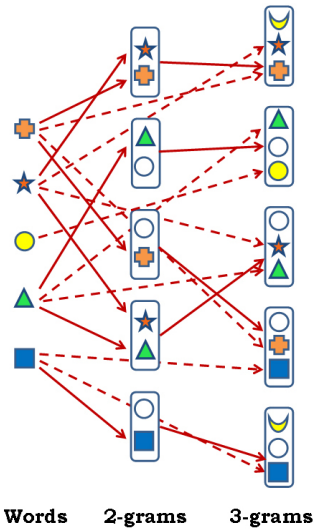


Fig. 3. Relations construction in a three-level hierarchy

3.3 Visualization of Network of Natural Term Hierarchy

We select top-20 Euromaidan terms (unigrams, bigrams and trigrams) with the largest CHVG weights to visualize the network we build. These terms are presented in Table 1.

Table 1. Top-20 Euromaidan terms with the largest CHVG weights

Unigram	Bigram	Trigram
Украина /Ukraine/	Виктор Янукович /Viktor Yanukovych/	президент Виктор Янукович /President Viktor Yanukovych/
Киев /Kiev/	центр Киева /Centre of Kiev/	сотрудники правоохранительных органов / law enforcement officials /
власть /Power/	верховная Рада /Verkhovna Rada/	введение чрезвычайного положения /Introduction of state of emergency/
страна /State/	улица Грушевского /Grushevskogo Street/	батькивщина Арсений Яценюк /Batkivshina Arseniy Yatsenyuk/
Янукович /Yanukovych/	президент Украины /President of Ukraine/	Олимпийские игры Сочи /Olympic Games Sochi/
Майдан /Maidan/	Майдан Независимости /Maidan Ne-zavisimosti (Independence Square) /	Глава Администрации Президента /Head of presidential administration/
люди /People/	партия регионов /The party of regions/	фракция партии регионов /The party of regions fraction/
милиция /Police/	пресс-служба /Press centre/	штаб национального сопротивления /National resistance

Unigram	Bigram	Trigram
		headquarters/
Беркут /Berkut/	Арсений Яценюк /Arseniy Yatsenyuk/	действие благодати Пресвятой /Holy Grace effect/
оппозиция /Opposition/	Михайловская Площадь /Mikhailovskaya Square/	Майдан Незалежности Киев /Maidan Nezalezhnosti Kiev /
президент /President/	лидеры оппозиции /Opposition leaders/	страницы социальных сетей /Social network pages/
Яценюк / Yatsenyuk/	разгон Евромайдана /Euromaidan dispersal/	УДАР Виталий Кличко /UDAR Vitali Klitschko/
украинский /Ukrainian/	объявление перемирия /Armistice announcement/	Германия Франция Великобритания /Germany France UK/
Евромайдан /Euromaidan/	Виталий Кличко /Vitali Klitschko/	улица Грушевского Киев /Grushevskogo Street Kiev/
штурм /Attack/	Майдан Незалежности /Maidan Nezalezhnosti /	офис партии регионов /Office of the party of regions/
акция /Act/	акция протеста /Act of protest/	михайловская площадь киев /Mikhailovskaya Square Kiev/
здание /Building/	правый сектор /Right Sector/	силовой разгон евромайдана /Military dispersal of Euromaidan/
активист /Activist/	огнестрельное оружие /Firearms/	беркут внутренние войска /Berkut the internal troops/
МВД /Ministry of Internal Affairs/	правоохранительные органы /Law machinery/	премьер николай азаров /Premiere Mykola Azarov/
площадь /Square/	штурм зачистка /Attack cleanup/	мирная акция протеста /Peaceful protest act/
улица /Street/	штурм майдана /Attack of Maidan/	здание верховной рады /Verkhovna Rada building/
Грушевского /Grushevskogo/	внутренние войска /The internal troops/	законная власть Украины /Ukraine's legitimate government/
лидер /Leader/	применение силы /Use of force/	лидер партии УДАР /Leader of UDAR party/

Finally when the terms hierarchy network is constructed, we visualize it using Gephi tool (https://gephi.org). To load the network into a database we represent it by an incidence matrix in ".csv" format.

4 Results

To illustrate the final network we present a small fragment of 20 terms (20+20+20 in total) in Fig. 4.

It can be noticed that the words in large print (Киев / Kiev, президент / president, Майдан / Maidan) in Fig. 4 are the topmost terms from Table 1. These words represent the nodes with the highest weights. Unigram nodes are connected with bigram and trigram nodes, and bigram nodes are connected with trigram ones. Arc thickness is proportional to the joint frequency of the terms (i.e., n-grams) it unites.

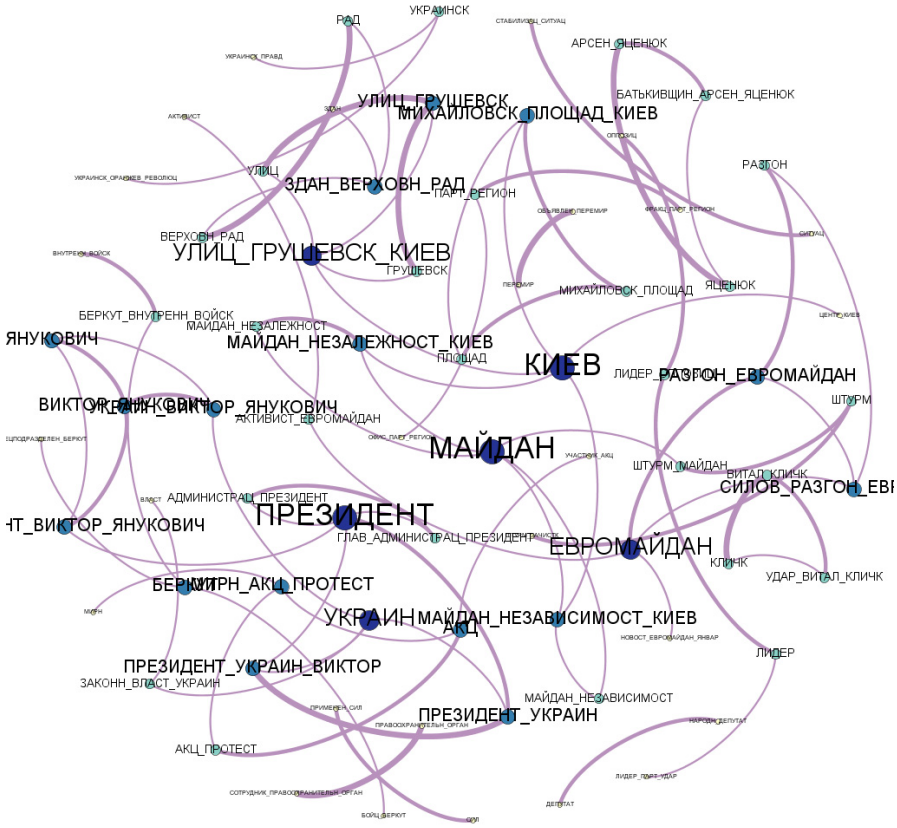


Fig. 4. Euromaidan Natural Network of Terms Hierarchy example (20+20+20)

We should also consider a larger Euromaidan network fragment (200+200+200), which is presented in Fig. 5. In Fig. 5 it can be seen that in spite of the large density of this fragment the terms “Киев” / Kiev and “Майдан” / Maidan remain in large print. Meanwhile the unigram “Президент” / President is replaced by the term “Беркут” / Berkut. It can be explained by the fact that the unigram “Беркут” / Berkut has higher weight than “Президент” / President.

After experimenting with networks of different sizes we deduced that node degree distribution (for outgoing links only) follows power law ($p(k) = Ck^{\alpha}$). It means that such networks are scale-free (see Fig. 4). Power coefficient α varies from 2.1 to 2.3 for networks of different sizes (e.g., from 20+20+20 to 500+500+500) that in general complies with Language Networks structure [0].

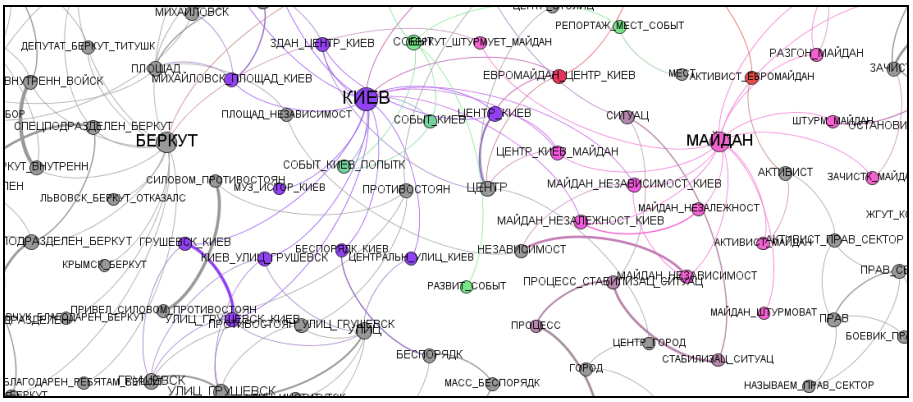


Fig. 5. Larger network fragment (200+200+200) visualized using Gephi

It also turned out that according to the proposed algorithm one node can have 5 ingoing links at most (for the network in our example, see Fig. 4). Single words (unigrams) have 0 ingoing links, bigrams – 2 ingoing links at most and trigrams – 5 ingoing links at most (with 3 links inherited from each word of a trigram and other 2 inherited from the two bigrams a trigram consists of). Top-20 nodes with the largest ingoing degrees for the 200+200+200 network of natural terms hierarchy are presented in Table 2.

Table 2. Top-20 nodes with the largest ingoing degree

Outgoing degree	Node
5	участники акции протеста /Protest act participants/
5	улица Грушевского Киев /Grushevskogo Street Kiev/
5	(президент) Украины Виктор Янукович /(President of) Ukraine Viktor Yanukovych/
5	силовой разгон Евромайдана /Military dispersal of Euromaidan/
5	мирная акция протеста /Peaceful protest act/
5	глава администрации президента /Head of presidential administration/
5	фракция партии регионов /The party of regions fraction/
5	бойцы спецподразделения Беркут /Berkut special units/
5	Батькивщина Арсений Яценюк /Batkivshina Arseniy Yatsenyuk/
4	администрация президента Украины /Ukrainian Presidential administration/
4	здание Верховной Рады /Verkhovna Rada building/
4	здания центра Киева /Buildings of the centre of Kiev/
4	Верховная Рада Украины /Verkhovna Rada of Ukraine/
4	УДАР Виталий Кличко /UDAR Vitali Klitschko/
4	сотрудники спецподразделения Беркут /Berkut officers/
4	сотрудники правоохранительных органов /Law machinery officers/
4	силовой разгон митингующих /Military dispersal of meeting participants/
4	политический кризис Украина /Political crisis Ukraine/
4	применение силы сторонами /Use of force by the parties/
4	пресс-служба МВД /Press centre of Ministry of Internal Affairs/

The nodes with the largest ingoing degree are also semantically the most important ones. They include the following word combinations: “участники акции протеста” /Protest act participants/; “улица Грушевского Киев” /Grushevskogo Street Kiev/; “силовой разгон Евромайдана” /Military dispersal of Euromaidan/; “мирная акция протеста” /Peaceful protest act/; “бойцы спецподразделения Беркут” /Berkut special units/.

CHVG values are calculated for single subjects as well, and the network is constructed for them. In Fig. 6 three network examples are shown. Their interrelation network is given in Fig. 7.

Our assumptions regarding the importance of the selected events for network constructing were confirmed during the experiments with informants. Each informant was given a standard instruction: “Remember the recent events in the world. Write down 10-15 words which are best to describe these events”. More than 40 informants were questioned [11].

In Table 3 the results of the experiments with informants are shown. In spite of the fact that the informants were not asked to describe the events in Ukraine, the majority of them still speak about the Euromaidan events.

Table 3. Significance of the selected events (results of public opinion poll)

%	Events. Informants under 30 ages	%	Events. Informants of 30 ages and older
39	The joining of Crimea to the Russian Federation	58	Winter Olympic Games in Sochi (Russia), the joining of Crimea to the Russian Federation
24	Disturbances in Maidan		
18	Olympic Games in Sochi	40	Disturbances in Maidan
	Excellent results of the Russian	33	Referendum in Crime
14	team in the Winter Olympic Games, Referendum in Crimea, Sanctions	23	Excellent results of the Russian team in Winter Olympic Games
	against Russia, Civil war in Ukraine	20	Civil war in Ukraine, Sanctions against Russia
12	Murders of civilian residents in Ukraine	13	Murders of civilian residents in Ukraine, War in the East of Ukraine
8	Beginning of combat operations in the Donetsk republic, Escape of Yanukovich from the country	8	Escape of Yanukovich from the country, Beginning of combat operations in the Donets republic, Revolution in Ukraine, Deceitful propaganda in Russian media, Little green men in Crimea, Crisis in Ukraine

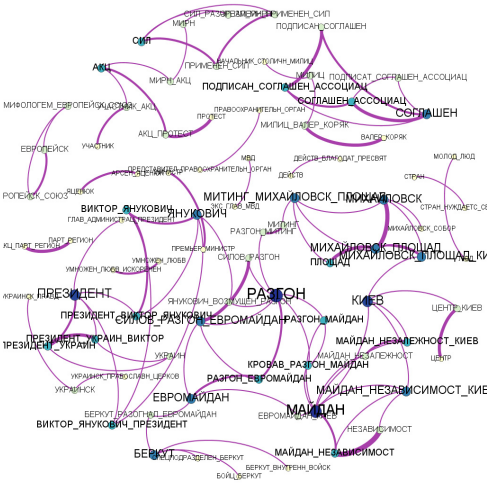
It is important to note that all the informants were divided into two groups according to their age. People of older age turn out to be quite critical while estimating the events of Euromaidan. On the other hand, one can find a large number of appraisal words in the answers of the younger group.

To confirm that the keywords and word combinations which we got as terms during the process of network construction are really semantically important for our theme we conducted another experiment with informants. We asked them to define the subject which these keywords can be connected with.

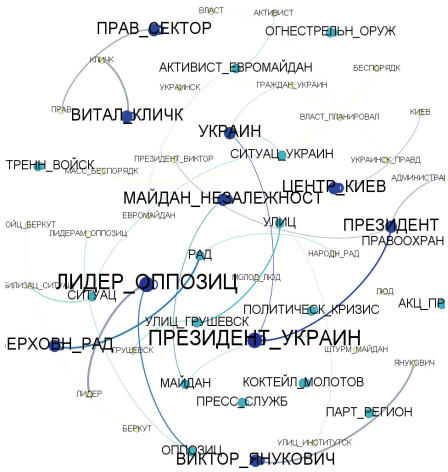
There were 7 informants in total and all of them were sure that these keywords were extracted from the texts about the events of Euromaidan in Kiev.

Such high level of agreement is caused by the high subject homogeneity of the corpus which we chose to analyze. In fact, all the texts within the corpus describe the same event.

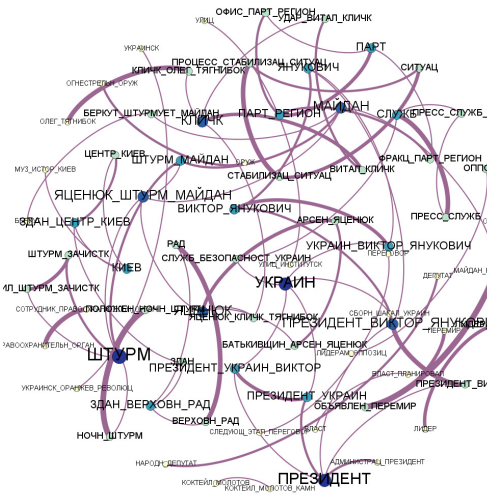
In Fig. 7 it is shown that a set of terms corresponds to each subject (a node identified with a date). The terms which take place on several different dates can be seen in the central part of the network while those which are more specific appear at the periphery.



a



b



c

Fig. 6. Euromaidan network (20+20+20) (a – 2013.11.30, b – 2014.01.22, c – 2014.02.19)

Central zone does not necessarily include all the terms from all the subjects – it is enough to include some portion of a subject's terms, e.g., one half. The more terms the central zone of a subject includes, the closer its content is to the main events trend, and the more relevant it is. In our example the "2014.01.22" node is the most relevant to the general events trend (see Fig. 7).

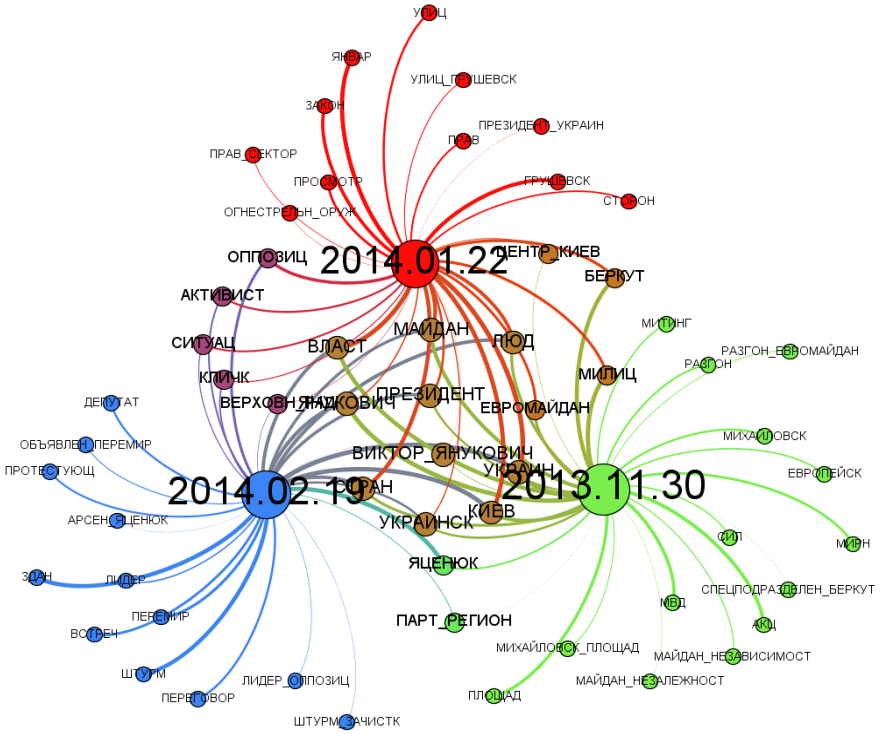


Fig. 7. Euromaidan terms interrelations for the three chosen dates

We also propose a linguistic criterion of subject relevance: the more terms of a paper are in the central zone of the term interrelation network the more relevant this subject is to the general events theme. In other words, subject relevance is proportional to the number of its terms in the central zone of the term interrelation network.

5 Conclusion

As a result of the research:

- an algorithm of constructing a network of natural terms hierarchy based on corpus analysis is proposed;
- the algorithm is illustrated with the examples of a Euromaidan-related network;
- network of natural terms hierarchy appears to be scale-free while considering outgoing links;

- programming tools for visualization of a network of natural terms hierarchy are introduced;
- the criterion of subject relevance to the event is proposed;
- the verification of this criterion according to the informants opinion is proposed.

Language network, constructed according to the proposed method, can be used as 1) a basis for ontology construction (e.g., for Ukrainian acts of protest theme), 2) a tool for database navigation and 3) a tool for organizing user prompts in information retrieval systems.

Our future work includes constructing networks on the base of a less homogeneous corpus. We are already working on the improvement of the estimation of our results, involving more informants with more complex stratification (country, region, profession and so on).

Acknowledgments

The authors acknowledge Saint-Petersburg State University for a research grant 30.38.305.2014. Special thanks to Vlad Kotov for sharing materials with informants.

References

Bolshakova, E.I., Klyshinsky, E.S., Lande, D.V., Noskov, A.A., Peskova, O.V., Yagunova, E.V. (2011) *Avtomaticheskaya Obrabotka Tekstov na Estestvennom Yazike i Kompuernaya Lingvistika*a. Moscow, MIEM, 272 p. Title in Russian: Автоматическая обработка текстов на естественном языке и компьютерная лингвистика.

Drymonas, E., Zervanou, K., Petrakis, Euripides G.M. (2010) Unsupervised Onthology Acquisition from Plain Texts: The OntoGain System. *Lecture Notes in Computer Science*, vol. 6177, pp. 277–287.

Lande, D. V., Snarskii, A. A. (2013) Compactified HVG for the Language Network. *Proceedings of the International Conference on Intelligent Information Systems*: The Conference is dedicated to the 50th anniversary of the Institute of Mathematics and Computer Science, 20–23 Aug. 2013, Chisinau, Moldova: Proceedings IIS / Institute of Mathematics and Computer Science, pp. 108–113.

Lande, D.V. (2014) Building of Networks of Natural Hierarchies of Terms Based on Analysis of Texts Corpora. E-preprint viXra 1404.0069.

Lande, D.V., Snarskii, A.A., Yagunova, E.V., Pronoza, E.V. (2013) The Use of Horizontal Visibility Graphs to Identify the Words that Define the Informational Structure of a Text. *Proceedings of the 12th Mexican International Conference on Artificial Intelligence*, pp. 209–215.

Liu H., Salerno J., Young M. Berlin (eds) (2009) *Social Computing and Behavioral Modeling*. Springer, 280 p.

Luque, B., Lacasa, L., Ballesteros F., Luque, J. (2009) Horizontal Visibility Graphs: Exact Results for Random Time Series. *Phys. Review E*, pp. 046103-1– 046103-11.

Poon, H., Domingos, P. (2010) Unsupervised Ontology Induction from Text. *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 296–305.

Wong, W., Liu, W. & Bennamoun, M. (2012) Ontology Learning from Text: A Look back and into the Future. *ACM Computing Surveys*, vol. 44 (4), pp. 20:1–20:36.

Yagunova, E., Lande, D. (2012) Dynamic Frequency Features as the Basis for the Structural Description of Diverse Linguistic Objects. *CEUR Workshop Proceedings*, vol. 934, Russian Conference on Digital Libraries, pp. 150–159.

Yagunova, E.V., Krilova, I.V, Makarova, O.E., Pivovarova, L.M. (2014) Snezhnaya revoluciya v Rossii: znachimie nominacii, sobitiya, ocenki (oceenka sobitiy informantami i dannie SMI). In: *"Mi ne nemi!": tvorchesvo protestuyshey ulici*, Tartu. Title in Russian: "Снежная революция в России": значимые номинации, события, оценки (оценка событий информантами и данные СМИ)

Lande D., Snarskii A., Jagunova E. Network of Natural Hierarchies of Terms of News Messages on the "Euromaydan" Events. *CEUR Workshop Proceedings*, vol. 1297, Russian Conference on Digital Libraries, pp. 55-74 (2014)

Stampato in Italia
presso LegoDigit s.r.l.
via Galileo Galilei, 15/1
38015 Lavis (TN)
luglio 2015