# Comparison pattern matching and creative simile recognition

**Vlad Niculae**
Université de Franche-Comté, Besançon
Center for Computational Linguistics, University of Bucharest
vlad@vene.ro

## Abstract

Comparisons are phrases that express the likeness of two entities. They are usually marked by linguistic patterns, among which the most discussed are *X is like Y* and *as X as Y*. We propose a simple slot-based dependency-driven description of such patterns that refines the phrase structure approach of Niculae and Yaneva (2013). We introduce a simple similarity-based approach that proves useful for measuring the degree of figurativeness of a comparison and therefore in simile (figurative comparison) identification. We propose an evaluation method for this task on the VUAMC metaphor corpus.

## 1 Introduction

The comparison structure is a common linguistic pattern used to express similitude or distinction of two entities with respect to a property. When the comparison is not intended to be taken literally, it is called a *simile*. Identifying comparison structures is important for information extraction, as it is a way of asserting properties of entities. The simile, on the other hand, is interesting for the striking creative images it often produces:

> "Mrs. Cratchit entered: flushed, but smiling proudly: with the pudding, like a speckled cannon-ball, so hard and firm, (...)" (In "A Christmas Carol" by Charles Dickens)

The simile, as a figure of speech, is receiving an increasing amount of interest, after being historically discarded as a less interesting form of metaphor. To clarify that the expressive span of the metaphor and the simile overlap but are different, Israel et al. (2004) gives examples of metaphors that cannot be perfectly transformed into similes, and vice versa. Further supporting this point, Hanks (2012) identifies many cases where the simile is used creatively as a way of describing things by constructing images that surpass the realm of the possible and the experienced.

## 2 Corpora

The VU Amsterdam Metaphor Corpus (Steen et al., 2010), henceforth VUAMC, is a subset of British National Corpus (BNC) Baby (Burnard, 1995) annotated for phenomena related to linguistic metaphor. It consists of 16 202 sentences and 236 423 tokens. About half (50.7%) of the sentences have at least an *mrw* (metaphor-related word) annotated. Of more interest for our study is the *mFlag* (metaphor flag) annotation, which surrounds trigger phrases for figurativeness. Table 1 shows the most frequent *mFlag* tags. We investigate the use of this annotation for automatically evaluating simile identification. Given the underrepresentation of similes in the VUAMC, we chose to only present experiments using the comparison patterns involving *like*. The methods used are not pattern specific. Up to a degree of variation given by subtle language behaviour, they should apply to any comparison, as they only involve the contents of the comparison constituents that will be described in section 3.

In addition to the VUAMC, we used the collection of examples from (Hanks, 2012) and a subset of extracted matches from the BNC (Burnard, 1995). All text was tagged and parsed using TurboParser (Martins et al., 2010) using the basic projective model and lemmatized using `Treex`[1] (Popel and Žabokrtskỳ, 2010).

## 3 Syntactic aspects

### 3.1 Characterisation of comparisons

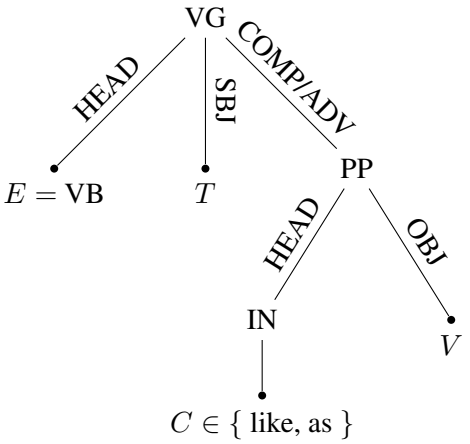
[1] `http://ufal.mff.cuni.cz/treex/index.html`

| flag | count | freq. per sentence |
|---|---|---|
| like | 57 | 0.35% |
| as | 28 | 0.17% |
| as if | 7 | 0.04% |
| of | 6 | 0.04% |
| *other* | 45 | 0.28% |
| total | 143 | 0.88% |

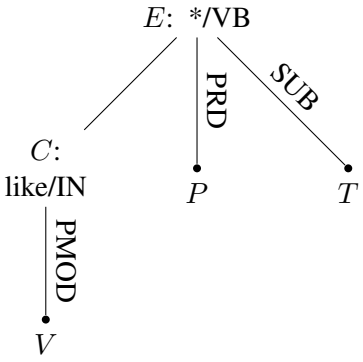
Table 1: Metaphor flags in VUAMC

```
dict(slot='E',
     pos=lambda x: x.startswith('VB'),
     kids=[
          dict(slot='C',
               form='like',
               pos='IN',
               kids=[dict(slot='V',
                     deprel='PMOD')]),
          dict(slot='T',
               deprel='SUB',
               optional=True),
          dict(slot='P',
               optional=True,
               deprel='PRD'),
      ])
```

Listing 1: Python code representing the simple pattern for comparisons using *like* defined by Figure 1b.




(a) GLARF-style pattern.




(b) DEP-style pattern. Its Python representation can be found in Listing 1.

Figure 1: Visualisation of the two types of approaches for encoding the *X is like Y* pattern.

Hanks (2012) identifies the following constituents of a comparison: the topic ($T$), the eventuality ($E$), the shared property ($P$), the comparator ($C$) and the vehicle ($V$). An example (adapted from the BNC) of a simile involving all of the above would be:

> [He $^{T}$] [looked $^{E}$] [like $^{C}$] [a broiled frog $^{V}$], [hunched $^{P}$] over his desk, grinning and satisfied.

Niculae and Yaneva (2013) used constituent parsing with GLARF (Meyers et al., 2001) transformations in order to match several hand-written comparison patterns. While the normalizations performed by GLARF allow for more general patterns (constructions using auxiliary verbs such as *have been* are handled transparently), the tool is only available in English, and it proves error-prone in practice for complex sentences.

Dependency parsing, based on the formalism of Tesnière and Fourquet (1959), has been more actively developed recently. Compared to constituent parsing (phrase-structure grammars), dependency trees are easier to annotate, hence the availability of dependency treebanks and trained models for more languages. The space of possible dependency trees of a sentence is much smaller than the space of possible constituent trees, allowing for better models. Recent work in structured prediction includes the TurboParser (Martins et al., 2010), which we use in this work.

Figure 1 shows the GLARF-style pattern for comparisons using *like*, along with a corresponding dependency pattern.

### 3.2 Encoding and matching dependency patterns

In the case of phrase-structure treebanks, the powerful tools *Tgrep* and *Tgrep2*[2] permit fast extraction of trees sharing common patterns. Unfortunately, their formalism is inappropriate for query-

[2] http://tedlab.mit.edu/~dr/Tgrep2/

ing dependency trees. Additionally, while expressive, their syntax is arguably opaque and unwelcoming. We propose a simpler pattern matcher written in Python with patterns represented as Python code. The resulting patterns look similar to their graphical representation. This representation is a step closer to automatic construction of patterns, compared to hand-written pattern matching using conditionals. The implementation is currently available in the *comparison-pattern* package [3] under the permissive 3-clause BSD license. Like *Tgrep* works on parenthesised trees, common for representing constituent parses, our matcher takes CoNLL-style input.

For brevity, we henceforth refer to our dependency pattern matching system as DEP.

Listing 1 shows the code needed to represent the pattern. For certain underspecified patterns with symmetries, it's possible that several matches with the same root occur. Our matcher returns all of them and choosing the appropriate one is left as a separate problem that we do not treat in this work.

### 3.3 Comparison identification results

On the examples from (Hanks, 2012), DEP improves recall by identifying 6 additional matches, while losing only 2, one due to a POS tagging error and the other due to a parsing error. In cases when both systems match a sentence, it is sometimes the case that DEP provides more complete information, especially in the case of convoluted sentence structures.

On the subset from the BNC used by Niculae and Yaneva (2013), we examine only the points of disagreement between systems (sentences matched by one but dismissed by the other). Even though this analysis ignores the possibility of making qualitatively different matches for the same sentence, we opted for it for convenience, as evaluation needs to be manual. Contrary to Niculae and Yaneva (2013), we disregard matches that don't identify the vehicle of the comparison, as we are interested in finding common vehicles, for mining different comparators.

At first sight, DEP identifies 199 sentences that GLARF misses, while GLARF matches 36 instances missed by DEP. Upon going through the examples, we find that 43 matches out of the 199 are spurious because of preventable tagging or parsing errors, many of them in transcribed speech, where the word *like* functions as a filler word. However, 11 out of the GLARF-only matches were also spurious. Using dependency parsing is therefore a net gain for comparison identification.

[3] http://github.com/vene/comparison-pattern

| System | $P$ | $R$ | $F_1$ | count |
|---|---|---|---|---|
| LEXBL | 0.166 | 1.00 | 0.284 | 320 |
| GLARF | **0.303** | 0.434 | 0.357 | 96 |
| DEP | 0.241 | **0.717** | 0.360 | 158 |
| DEPSEM | 0.252 | **0.717** | **0.373** | 151 |

Table 2: Simile identification performance, with respect to the 53 instances of *mFlag=like* annotation in VUAMC. LEXBL is the baseline that retrieves all sentences that contain the preposition *like*. The last column measures the number of retrieved instances.

### 3.4 Automatically evaluating simile retrieval

On VUAMC, we can use the *mFlag* annotation as ground truth for evaluating pattern matching. However, as the focus of the corpus is figurative language, literal comparison are not marked. Because pattern matching finds comparisons, without any semantic processing, the retrieval precision will be low (all literal comparisons would be seen as spurious simile matches). However, it passes the sanity check against the LEXBL baseline that simply returns all sentences containing the preposition *like* (after part-of-speech tagging). To our knowledge this is the first attempt at an automatic evaluation of simile retrieval performance. The results are presented in table 2. Even though raw extraction $F_1$ score is very close, DEP has much better recall and therefore leaves more way for improvement with semantic methods, as promised by our DEPSEM heuristic described in section 4.1. This heuristic manages to improve precision at no cost in recall.

## 4 Semantic approximations of figurativeness and creativeness

### 4.1 Approach

Though the setting imposed by the VUAMC annotation is to distinguish figurative from literal comparisons, the problem is much more nuanced. In addition, there exists a distinction between conventional and creative language, as discussed for example in (Hanks, 2013). We investigate the use of language conventionality as a proxy to negative

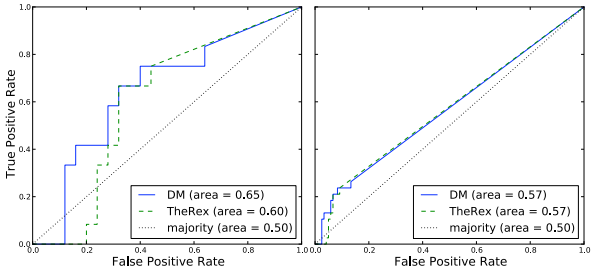



Figure 2: ROC curves for similarity as a predictor of comparison figurativeness measured on comparisons found in the semantic resources (left) and on all comparisons, assuming missing values are zero (right).

figurativeness. We approximate conventionality as similarity between the tagged lemmas of the head words of $T$ and $V$. To this end, we make use of two precomputed, freely accessible resources. The DEPSEM heuristic filters out matched comparisons with similarity scores above a manually-set threshold, under the assumption that comparisons against highly similar things are unlikely to be figurative.

### 4.2 Resources

**Distributional Memory** (Baroni and Lenci, 2010), henceforth DM, is a general-purpose model of corpus-driven semantics. While it comes in a tensor form of *word-link-word*, we use the distributional word representations induced by random indexing, available online[4]. Shutova et al. (2010) used distributional verb-noun clusters in metaphor identification, suggesting that such methods can be adopted for measuring figurativeness. We measure similarity as the cosine between word vectors.

**Thesaurus Rex** (Veale and Li, 2013), henceforth THEREX[5] is a knowledge base of categories and stereotypes mined from the web using the patterns *as X as Y* and *X such as Y*. While less complete than a knowledge-cheaper distributional resource such as DM, THEREX contains structures that can be explored for simile simplification, by inferring the missing $P$ as discussed in (Niculae and Yaneva, 2013). We measure similarity between noun pairs as a sum of the weights of all shared categories of the two words and categories of each of the word, derived from the other[6].

[4] http://clic.cimec.unitn.it/dm/

[5] http://ngrams.ucd.ie/therex2/

[6] This strategy proved better than measuring just the shared categories, or than simply counting instead of adding weights. For brevity we omit the particular results.

### 4.3 Evaluation

Figure 2 shows the ROC curves for the two methods, where the similarity scores are seen as predictors of the target variable that indicates whether *mFlag=like* is annotated within the sentence. It can be seen that these measures perform better than the baseline of always choosing the majority class.

For a qualitative evaluation and proof of concept, we point out several comparisons with low and high similarities, identified in the BNC.

> The piano ripples like patent leather. [DM(piano, leather) $= 0.076$]
>
> This is a vivid and funny production and their expertise makes the intricate puppetry go like a dream. [DM(puppetry, dream) $= 0.076$]
>
> Ink, like paint, uses subtractive colour mixing while the video monitor uses the additive colours; red, green and blue, to produce the same effect. [DM(ink, paint) $= 0.502$]

## 5 Conclusions and future work

We improve upon previous work in comparison pattern matching by using dependency parsing, and at the same time provide a more general interpretation of pattern matching. Our approach is much easier to adapt to other languages, as it needs a POS tagger and a dependency parser.

We show that there exists some negative correlation between lexical semantic similarity and figurativeness, that we exploit in a simple heuristic for simile identification. Such measures can be used as features for simile classifiers.

Obvious improvements include measuring similarity between children nodes and not just the head node of each argument, or measuring other arguments ($E$ and $V$, for example). The shared categories of pairs of nouns and poetic categories of single nouns available in THEREX show promise for simile simplification. Measures of compositionality in distributional semantics as used by Vecchi et al. (2011) for identifying impossible constructions are expected to be better suited for our task than the representation based on simple co-occurrences.

## References


Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Lou Burnard. 1995. *British National Corpus: Users Reference Guide British National Corpus Version 1.0*. Oxford Univ. Computing Service.

Patrick Hanks. 2012. The Roles and Structure of Comparisons, Similes, and Metaphors in Natural Language (An Analogical System). In *Presented at the Stockholm Metaphor Festival*, September 6-8.

Patrick Hanks. 2013. *Lexical Analysis: Norms and Exploitations*. Mit Press.

Michael Israel, Jennifer Riddle Harding, and Vera Tobin. 2004. On simile. *Language, Culture, and Mind. CSLI Publications*.

André FT Martins, Noah A Smith, Eric P Xing, Pedro MQ Aguiar, and Mário AT Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 34–44. Association for Computational Linguistics.

Adam Meyers, Ralph Grishman, Michiko Kosaka, and Shubin Zhao. 2001. Covering treebanks with glarf. In *Proceedings of the ACL 2001 Workshop on Sharing Tools and Resources - Volume 15*, STAR '01, pages 51–58, Stroudsburg, PA, USA. Association for Computational Linguistics.

Vlad Niculae and Victoria Yaneva. 2013. Computational considerations of comparisons and similes. In *51st Annual Meeting of the Association for Computational Linguistics Proceedings of the Student Research Workshop*, pages 89–95, Sofia, Bulgaria, August. Association for Computational Linguistics.

Martin Popel and Zdeněk Žabokrtskỳ. 2010. Tectomt: modular nlp framework. In *Advances in Natural Language Processing*, pages 293–304. Springer.

Ekaterina Shutova, Lin Sun, and Anna Korhonen. 2010. Metaphor identification using verb and noun clustering. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1002–1010. Association for Computational Linguistics.

G.J. Steen, A.G. Dorst, and J.B. Herrmann. 2010. *A Method for Linguistic Metaphor Identification: From Mip to Mipvu*. Converging evidence in language and communication research. Benjamins.

Lucien Tesnière and Jean Fourquet. 1959. *Eléments de syntaxe structurale*, volume 1965. Klincksieck Paris.

Tony Veale and Guofu Li. 2013. Creating similarity: Lateral thinking for vertical similarity judgments. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 660–670, Sofia, Bulgaria, August. Association for Computational Linguistics.

Eva Maria Vecchi, Marco Baroni, and Roberto Zamparelli. 2011. (linear) maps of the impossible: capturing semantic anomalies in distributional space. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 1–9. Association for Computational Linguistics.