KBC: THE KNOWLEDGE BUILD COMPLEXITY

Trevor Wine

Abstract. A knowledge-based complexity measure is introduced for assessing the complexity of a given entity. At its core, the measure relies on reducing the entity to a sum of human knowledge inputs. A knowledge baseline is first established, below which the knowledge levels are considered negligible. A complexity tree is then derived, with complexity nodes defined by an 'is a component'-type relationship, and each branch terminating when it arrives at the knowledge baseline. Different trees may be partially compared by summing the complexity along all branches, or, more fully, by considering the tree structure and individual complexities at each node. The paper concludes with an example in practical use, a routine which receives an entity name and performs a basic recursive knowledge search with a specialized web crawler on the Wikipedia corpus. To help determine relevance for recursion, the NLP BERT machine learning model is used for relation extraction.

1. Introduction

Methods for measuring the complexity of an arbitrary object or system is an active area of research, particularly where business stakeholders see financial benefit. Correspondingly, the extant literature on entity complexity tends to skew toward manufacturing processes, such as factory or supply chain operations and oversight, with frequent overlap in the areas of systems-, product engineering, and system design [5]. Most measures attempt to capture a broad range of complex behavior: from design, construction, and management, to unintended emergent effects [15, 13].

The literature details various approaches in the assessment of complexity in production contexts. Commonalities include identifying primary subdivisions of interest, notably manufacturing, supply chains, design, and corporate organization [2]. More specifically, dimensions of complexity in the manufacturing process itself may include product structure, the number of stages or tools required in manufacture of the product, the structure of the shop or plant, the depth and breadth of planning and scheduling functions for operations, as well as the number of products produced by a facility, the number of manufactured items in the completed products' bill of materials (BOM), and the degree of part commonality (an expression of the proportion of parts that are used by multiple end products) [8, 6, 7].

To derive quantitative values for complexity in these manufacturing dimensions, well-established complexity measurements are often utilized. Measures of algorithmic complexity, for instance, may be applied to planning and scheduling functions. Entropy-based measures have been applied to machinery, operations depth, and product routing [19, 1], as well as supply chains [9]. Network, or graph, complexity is another useful metric, which has been applied generally to human-engineered systems [5].

At broader scope, macroeconomic measures of product complexity have also been developed. The economic complexity index (ECI), is directly related to the quantities of complex product a country produces [10, 11]. The product complexity index (PCI), is an indirect measure of the knowledge level associated with a product, measured internationally by considering comparative advantage.

More general measures of entity complexity tend to grow sparse, partly perhaps for lack of stakeholders with an interest, and partly because of obvious difficulties in creating an all-inclusive measure. Seth Lloyd has offered three main qualities for a more universal complexity: difficulty of description; difficulty of creation; and degree of organization [14].

This article proposes a measure that is narrower in scope, but offers advantages in computation and allows comparisons between output of highly divergent types. The measure reflects the cost in knowledge and intellect to build an item up from a global developmental baseline.

Key words and phrases. econometrics, economic complexity theory.

2. Considerations and key elements of the KBC

We define the knowledge build complexity (KBC) as the recursive totality of knowledge and intellect required to build an item from the baseline state. In its purest form, the KBC will be encoded in a componentization tree. See Figure 1.

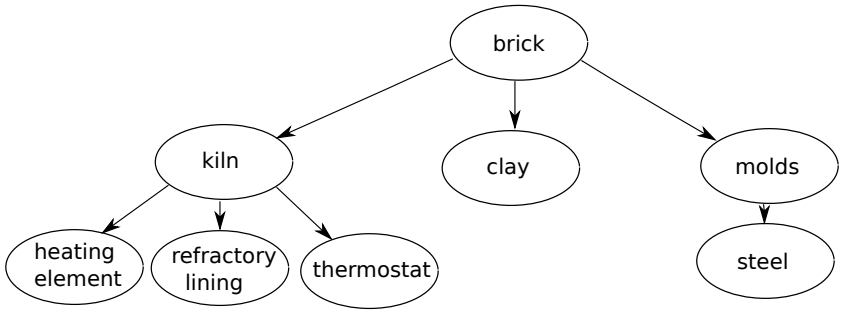

Figure 1. *The first few levels of simplified KBC componentization tree for entity 'brick.'*

We define knowledge as "an ideational (i.e. conceptual rather than physical) construct generated through the agency of the human mind" [12]. We define a knowledge process as the cognitive effort of an individual, for instance a subject matter- or domain expert, working over some period of time.

For this article, only human knowledge processes will be considered, but the definition may be usefully extended to other forms. For example, the 'construction' of a biological organism may be credited to the very basic knowledge process of Darwinian evolution working over a very long time-frame (hundreds of millions, or even billions of years). And with the advent of artificial intelligence, certain computational processes are beginning to rival human knowledge processes: for example, generational adversarial networks (GAN) for novel image creation, or generative pre-trained transformers (GPT-3) in the realm of natural language creation. Though these do rely on a vast corpus of already-generated human knowledge products (images and text), and are therefore at least partly measurable by KBC, it is conceivable that in the future AI may increasingly leave human generated content behind.

The KBC regards human knowledge processes as primary in the creation of an entity, viewing human knowledge as a fundamental resource that has required time and effort to build and accumulate. The more and higher degree of human knowledge inputs in the creation of an entity or system, the more complex it is assumed to be under this measure.

To help motivate this approach, human knowledge inputs often track with intuitive notions of the complexity of a system. A complex supply chain for example often requires more minds for its creation and oversight, likewise for a more complex factory space, or suite of components required to assemble a product. We may also consider Lloyd's three principal categories for entity complexity: difficulty of description; difficulty of creation; and degree of organization. Human knowledge processes will naturally be involved with or inclined toward higher levels in all three factors.

The KBC offers several other advantages over existing techniques. The reduction to a single consideration, that of knowledge processes, streamlines the measurement approach; this versus having to juggle entropy measures, algorithmic complexity, network complexity, etc. The simplicity of the reduction also allows for easy comparisons between widely different contexts–for example, comparing a microchip and a novel. The reduction also offers the potential to be, at least partly, automated, via data mining techniques. An example is given in the last section. Finally, with its universality, the KBC may be useful for a detailed assessment of aggregate complexity at varying degrees of scope–of everything from a factory, to some representative basket of products from a given country.

Where the KBC may encounter limitations is in the area of emergent effects. It is possible to design a system with relatively simple components that has, perhaps unintentionally, an emergent complexity. Inadvertent recursion or feedback processes may exist in a manufacturing process or supply chains that constitute an 'unmanaged' complexity (separate agents focused on near-details, which, in aggregate, produce complexity that none of the agents has an awareness of or control over). If the complexity of a system largely

arises from unforeseen complex interactions between fully understood components, then the KBC may fail to account for this. Of potential rescue is the likelihood that where there are emergent effects, human knowledge is usually not far away.

3. The componentization tree

Every entity under measurement with the KBC has a topology, that of the componentization tree. A portion of a simplified tree for a typical building element, a brick, is shown in Figure 1. In its fullest form, each node of the tree is also assigned a knowledge value subtree, and a KBC(assembly) subtree, which will be discussed in the next section.

The principal elements of the componentization tree are the recursive component levels for the entity in question. The tree is a directed rooted tree, with the principal entity at the root and components essential to the creation of the entity at progressively lower levels. The direction refers to a general 'is a component of' relationship (either as a true physical part, or in the construction of–that which at least must be present and available before its progeny at higher levels in the tree can be created). The convention for this article is to place the root uppermost (or leftmost) on the page.

It is possible for a component to appear more than once in the tree. For instance, a steel mill may feature a large amount of steel equipment on the factory floor and in the construction of the factory itself. A more problematic case might involve two different entities that include each other in their componentization trees, leaving the potential for endless recursion. The procedure in such a case will be to retain the highest node in the tree (closest to the root) where the component first occurs. Lower appearances of the node may be shown, but will be considered parenthetical and will not count toward the total complexity score. Such parenthetical appearances become important if we consider joining a subtree to another tree, where the component otherwise would not have a node in the subtree, nor have a node higher in the tree to which it is being joined.

The tree of Figure 1 is in reduced form. It ignores what could be important external components mentioned in the introduction, such as supply chains and corporate organization for business structures. It also stops the recursion after only two steps. The components of the kiln would need to be recursed on, for instance, along with the factory structure, to include the components used in the construction of the factory (i.e. the building that constitutes the factory, absent any equipment on the floor). It may be reasonable to assume some standard limit to recursion depth, and possibly set aside certain components as standard, such as the factory structure. These considerations will be discussed under the topic of occlusion.

Another essential consideration to tree construction is the establishment of a baseline state. A typical, useful example of a baseline state is preindustrial earth. The human knowledge inputs required to produce an entity below the baseline state are assumed to be negligible. When the complexity tree proceeds to a node that is within the baseline state, that branch of the tree stops. In the example tree of Figure 1, the component of clay is obviously available on preindustrial earth, and so would not be recursed on. However, the clay (or any other mineral or natural resource in this vein) might nonetheless require knowledge processes for its discovery, extraction, and transport to the factory site, which must be accounted for in the total complexity score, under KBC(assembly) as we'll see.

It remains to assign a complexity score to each node of the tree.

4. General formulas

General formulas are presented for determining the complexity of an entity.

Knowledge value:

The knowledge value (KV) quantifies the work of relevant knowledge processes. The elements in the knowledge value are,

$$KV : g(KP, t), \mathcal{G}(\text{research}) \tag{1}$$

where,

- g is an arbitrary function;
- KP represents the knowledge processes of consequence;
- t is the time the knowledge processes work over;

· $\mathcal{G}$(research) is the componentization subtree of everything used in research efforts (e.g. measurement devices, other research papers, etc.).

Note, in a slight abuse of terms we may refer to knowledge value as $h(g(KP,t), f(\mathcal{G}(\text{research}))$, i.e. as a numerical value obtained from some functional composition of $g(KP,t)$ and a function g acting on research tree, $\mathcal{G}$(research). This simplifies the KV, swapping the combination of $g(KP,t)$ and the full research tree for a numeric value.

Knowledge build complexity:

The total knowledge build complexity (KBC) for the entity is a function of the entity's full componentization graph:

$$KBC(\text{entity}) = f(\mathcal{G}(\text{entity})),$$

where $f : \mathcal{G} \to \mathcal{R}^n$ is some appropriate function on graphs that returns a vector (or scalar) value.

Let KV(organization) at a node represent the total knowledge processes and research components required to conceptualize the whole of the entity as the sum of its components, as well as perform any ongoing oversight of the production process.

Let $\mathcal{G}$(assembly) at a node be the componentization subtree associated with all components required to assemble the entity represented in that node as the sum of its component parts. For example, if the entity is a factory, then the first-degree component parts may be machines $1, ..., k$ on the factory floor. Then $\mathcal{G}$(assembly) would amount to all components required to transport the machinery to the factory and assemble it on the factory floor.

To create the full componentization graph, at each node we need to include the KV(organization), and $\mathcal{G}$(assembly). Each node, u, in the full componentization graph then follows the recursion relation,

$$u \longleftrightarrow (KV(\text{organization}); \mathcal{G}(\text{assembly})), \{\mathcal{G}(\text{component 1}), ..., \mathcal{G}(\text{component n})\}, \tag{2}$$

where,

- the node is labeled by the value $g(KP,t)$, which is the knowledge value corresponding to conceptualizing the whole of the entity as the sum of its components, as well as ongoing oversight of the production process;
- $\mathcal{G}$(component i) is the componentization subtree corresponding to essential component i; the associated directed edge is (u, c_i);
- $\mathcal{G}$(assembly) is associated with the KV(organization); $\mathcal{G}$(assembly) is the componentization graph of all tools, equipment, transport, etc. required to bring all components for the node together and assemble them; this includes supply chains and their management.

See Figure 2 for a complete diagram of a single node, based on formulas (1) and (2).

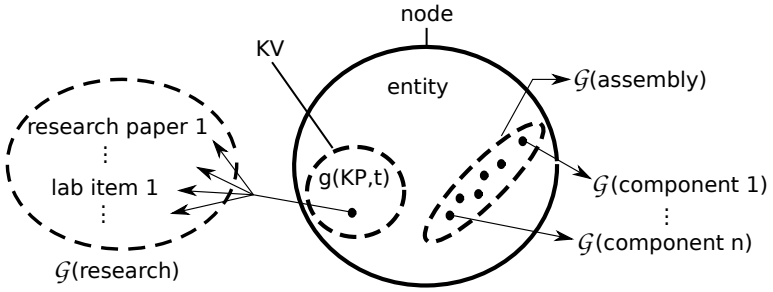

FIGURE 2. *Complete KV and subtree diagram for a single node in a componentization tree*

Note that in simpler forms of KBC, the componentization graphs may omit depiction of the $\mathcal{G}$(research) and $\mathcal{G}$(assembly) subtrees at each node, leaving the principal components of concern those that go directly into assembling the entity represented at each node (as in Figure 1).

Measuring KV:

The only values appearing in an entity's componentization tree are the knowledge process node values, $g(KP, t)$. Assigning a quantity to the knowledge value lies at the heart of the KBC measurement process.

Measurement of the knowledge process of equation (1) involves quantifying the relevant human knowledge inputs. For example, the routing of parts on the factory floor may have a systems engineer or other subject matter expert (SME) devote time to analysis, consulting with others and utilizing a number of essential papers and perhaps lab equipment in their work. The systems engineer would work over some time, t, and make use of the measurement devices and research of others in the literature, both of which which would have their own componentization graphs. Consultation with other experts would involve those experts' KP values as well, each working over some time period and perhaps also utilizing research material, to which further componentization would apply.

Full componentization may be difficult or very time consuming. In practical application, one might assign an average KP to SMEs in the field, and compute $g(KP, t)$ as simply,

$$g(KP, t) = \sum_i KP_{\text{avg}} * t_i,$$

where t_i is the time spent by SME i on entity conception. Further, one might assign an average KV to research papers, books, and patents in the field, and an average KV to research equipment. To then simplify the KV-associated componentization graph, $\mathcal{G}(\text{research})$, only the first few recursion levels of research materials could be considered. The truncated graph may then be converted to a value by summing the respective nodal values:

$$f(\mathcal{G}(\text{research})) \approx \sum_i KV_{\text{avg}}(i),$$

where $KV_{\text{avg}}(i)$ is the average KV of the ith research component, and we've truncated at the first recursion level.

The simplified KV-associated $\{g(KP, t), \mathcal{G}(\text{research})\}$ pair may then be consolidated into the single value,

$$KV_{\text{node}} \approx \sum_i KP_{\text{avg}} * t_i + \sum_i KV_{\text{avg}}(i). \tag{3}$$

Weighting componentization trees and occlusion:

In comparing complexity trees, there are several options. At its simplest, the sum of all KV nodal values in each complexity tree may be taken, and the two values compared. At a higher level of detail, one might first identify major subtree regions. For instance, in the factories under consideration there might be a raw materials section, a main production section, and a finishing section. Taking the summed complexity of each subtree as a component in a vector, vector norms may then be employed.

In general when comparing full complexity trees, there is a risk of exhaustive recursion rendering the KBC relatively meaningless. Specifically, if the recursion is done exhaustively without any attenuation or weighting, there is some risk that some number of entities link, by some finite degree of separation, to a very large number of other entities, making the KBC less useful–all such entities could have the same, very large, complexity value, especially if we rely on simple summing of the KV over all nodes. To retain value in the KBC for making meaningful comparisons, it's usually advisable to utilize some degree of occlusion.

For example, when comparing a brick to a stapler, the enclosing factory structure (this would include the outer factory building, the factory's physical plant, administrative offices–those elements peripheral to the specifics of the entity being made, but common across many production facilities) may be very complex, especially relative to the actual production components on the factory floor. For instance, an iPad used as an essential component in the offices of the brick factory may represent far more KBC than all the equipment on the factory floor combined. In consequence, the peripheral elements risk swamping out the more essential components in entity creation.

In full occlusion, we mask or ignore portions of the full complexity tree. For a meaningful comparison between the brick and stapler, for example, we might assume the supply chains and basic factory setups are the same for both. We correspondingly re-normalize by occluding (removing from consideration) the outer factory structure and supply chains from the respective trees (or simply not include them to begin with). Another possibility is truncating the tree at some fixed recursion depth.

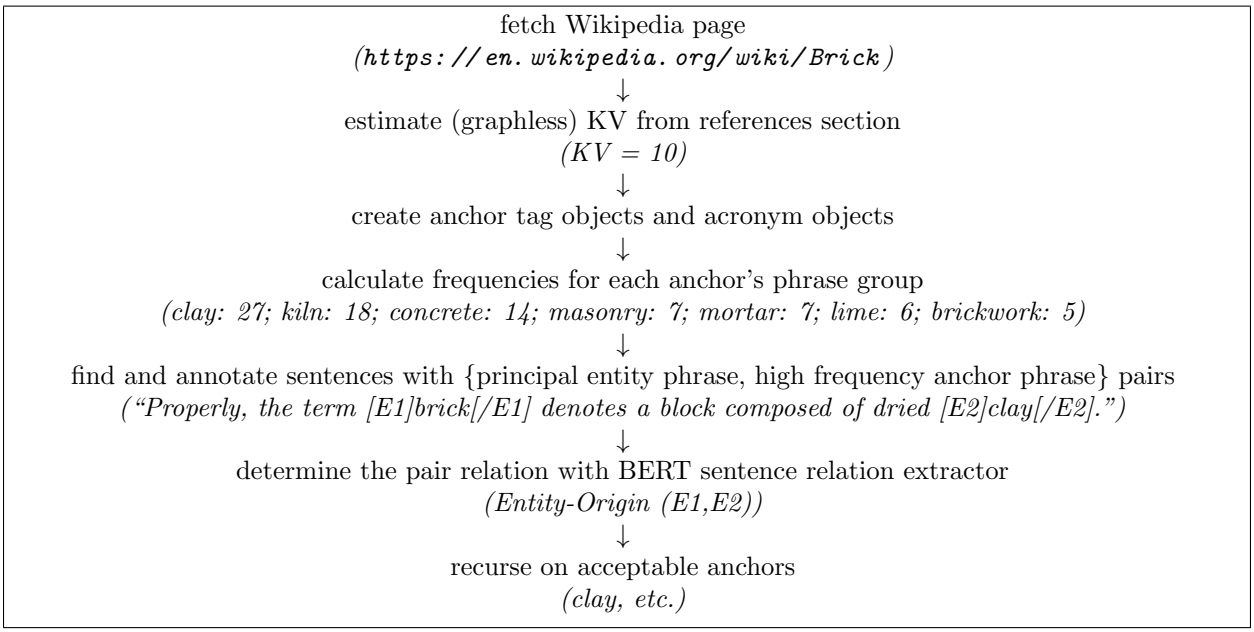

FIGURE 3. Program flow

Partial occlusion, or weighting, may also be applied. Appropriate weights can be assigned to nodes or subtrees in the complexity tree. For example, one might consider that the contribution of a rarely manufactured item to the total production complexity of a factory should be less than a commonly produced item [8]. Those aspects of the production facility and raw materials inputs would be weighted correspondingly lower.

There are of course cases where occlusion may not be desirable at all. For example, in comparing a mudbrick to a microprocessor the factory structure complexity and supply chains become highly relevant. In any case, it is important that the occlusion be uniformly applied, and noted when citing the relative complexities.

5. Python prototype

An advantage of the KBC's reduction of all essential inputs to a knowledge measurement is it renders the measure favorable to data mining. Rough KBC scores for common entities with publicly available development methods can be derived from specialized web crawlers, taking advantage of the abundant online availability of e.g. systems process descriptions and scholarship links.

A specialized Wikipedia crawler has been developed in the Python programming language to produce a KBC score for a given entity (article name) input. As of this writing, the code can be found at `https://github.com/tmwine/knowledge-build-complexity`. The procedural flow is shown in Figure 3.

After fetching the entity's Wikipedia page, the html text is processed.

Computing a simplified KV score:

To assign a simplified KV score to the article node in the spirit of equation (3), the article's references section is searched. A subroutine attempts to identify books, journal articles, and patents. Each instance receives one point. The article KV simply amounts to the point total. We do not attempt to compute individual knowledge process contributions (the $g(KP, t)$ values of formula (1)), nor the componentization graphs of research equipment, or other lower research material dependencies.

Determining high frequency anchor tags:

To parse the article text body, we use natural language processing (NLP). The core procedure involves identifying high frequency n-grams. Using a part of speech (POS) tagger, we first dismiss named entities

(e.g. Western Digital Corporation, or Topeka, Kansas) and proper nouns (John Smith), since these will be of little direct use in principal entity componentization. Next, since initial tests showed anchor tags almost exclusively carried all relevant information for componentization, the text is scraped for intra-Wikipedia anchor tags (anchor tags that refer to other Wikipedia articles; we do not go "outside" Wikipedia in the anchor tag search). We retain all anchor tags with phrase groups above some frequency cutoff.

The text is also scraped for acronyms. This adds complexity to the routine but was deemed essential for some articles. For example, an anchor tag with the phrase "metal-oxide-semiconductor" may be the only instance of that phrase in the article. The remaining instances are the associated acronym, "MOS." We will miss the potentially high frequency of that anchor tag if we don't associate the acronym with it, and merge "MOS" with the rest of the anchor tag's phrase group. Another challenge is plural forms. Both "kiln" and "kilns" must be treated as the same entity, and both must be included in the relevant anchor tag's phrase group for correct frequency counting. To discover singular / plural associations, the last word in each anchor tag phrase is run through a singular / plural complement creation routine, and the complementary phrase, if it exists, is searched for in the article text as well.

Other challenges include identifying single-word named entities or proper nouns that began sentences. This is partly dependent on the accuracy of the POS tagger, which both can make internal errors and simply be faced with too little information to make an accurate assessment. Wikipedia articles may also have a number of redirects–i.e. different URLs pointing to the same page. For example, `https://en.wikipedia.org/wiki/Integrated_circuit` and `https://en.wikipedia.org/wiki/Integrated_circuits` (note the plural) both point to the same page–if a single article contains both anchor tags, they need to be merged, otherwise the frequency score for the anchor reference in question will be artificially lowered.

Performing the relation extraction:

Once the anchor tag frequency list is generated, the highest scores are retained. For most articles, a cutoff of greater than or equal to 10 was deemed sufficient. It remains to determine whether a given high-frequency anchor tag is worth recursing on as a component. Indiscriminately recursing on anchor tags risks unwanted bloat, where the container-contained relation is not preserved, or, worse, done in reverse. For example, when considering "microprocessor," the anchor tag "transistor" should be recursed on, since a transistor is an essential component in the creation of a microprocessor. On the other hand, the anchor tag "computer" is not an essential component, but rather the whole to "microprocessor"'s part. Falsely recursing on "computer" could lead to a lot of unwanted complexity, compounded indefinitely if a similar mistake is made when considering what anchor tags to recurse on in the "computer" article, etc.

To prevent or reduce recursion bloat, we employ a natural language relation extraction model to categorize the anchor tags by their phrase groups. Relation extraction models attempt to determine the nature of the relationship, if any, between two language entities in a block of text. The entities may be named entities, proper nouns, or ordinary nouns and pronouns. For example, given the sentence, "A paper clip is a device used to hold sheets of paper together, usually made of steel wire bent to a looped shape." the relation extractor might attempt to determine the relationship between the entities "paper clip" and "wire." A correct result might be "component-whole," where "wire" is identified as a component to the "paper clip"'s whole.

The model employed is derived from that of Soares et al. [18]. Their Bidirectional Encoder Representations from Transformers (BERT)-based model [3] utilizes a matching-the-blanks technique to eliminate the need for human labeling of entity pairs in an unsupervised learning phase, allowing for much larger training sets. The model has been shown to have better accuracy than prior, prevailing methods. For a third-party version of the model, what has been used in this paper, see `https://github.com/plkmo/BERT-Relation-Extraction`. We employed a partly-trained version. Fuller training, on a larger text corpus, would further improve the relation extraction results.

The article's sentences are searched for simultaneous occurrences of both a phrase from the principal entity's phrase group, and the phrase group of a high-frequency anchor tag. For every such occurrence, the relevant sentence is tagged with markers to alert the model to the two entities of interest (see relevant step in Figure 3).

Pertinent relations from this model's 9 relational pairs are: Content-Container; Product-Producer; Entity-Origin; Member-Collection; Component-Whole; Instrument-Agency; Entity-Destination. In the not infrequent case there are multiple sentences containing the phrase pair, a form of consensus polling is used. A recursion decision is made on the basis of the result: most notably, if the anchor phrase in question satisfies a contained/container relation to the principal entity, or a producer/product relation, then recursion will occur.

Concluding example:

To serve as a concluding example, we consider two entities for comparison: a paper clip, and a microprocessor. The Python routine was run to a recursion depth of three: principal entity, first component list, second component list. The total entity score was derived simply from summing the simplified nodal KV scores of all components in all three layers. Shown in Figure 4 are the results for "paper clip," and in Figure 5 are the results for "microprocessor." Note, the Python routine and relation extractor did not pick up on "steel" when recursing on "wire." We added "steel" by hand. The complexity scores were,

$$KBC(\text{paper clip}) = 52,$$

$$KBC(\text{microprocessor}) = 504.$$

It's also important to note the Python routine did not pick up on several major components of integrated circuit fabrication. Fab plants have very high fixed-cost inputs, and contain correspondingly highly complex machinery. If such complexities were better accounted for and recursed on, the KBC scores would likely diverge more sharply. The recursion depth cutoffs also create some asymmetry in the scoring: for example, recursing far enough in "microprocessor," we would eventually encounter "steel," for instance in factory equipment, further separating the scores.

FIGURE 4. *Simplified componentitization tree for "paper clip," based on Python routine*

6. Discussion

This article has demonstrated an alternative to existing measures for computing entity complexity. A reduction to human knowledge inputs above a given baseline offers a single, quantifiable basis for the measure. Previous measures often employed a variety of techniques that may be very well suited to a particular aspect of the manufacturing process, but may be difficult to generalize to entities of arbitrary types. The KBC can nonetheless be useful to businesses, as another way to assess build- and production complexity. For governments, the KBC may be useful for highlighting areas more vulnerable to disruption, under the assumption that more complex processes and equipment are more vulnerable to failure under system shocks.

A working prototype of an automated KBC routine was offered in the form of a specialized Python web scraping tool. Several additional improvements could be made to the routine. Wikipedia was used because it is comprehensive and scrapeable, but including other sources, such as other encyclopedias or search engine links to relevant material would make componentization results much more robust. Another area of useful componentization information would be patent databases. The relation extraction routine could also be made more robust, or even augmented or replaced with more sophisticated NLP tools for accurately determining componentization.

The KV scoring could also be improved, with more thorough web searches for related scholarship, and might take into account a paper or journal's citation ranking, as well as a ranking score for specific SMEs. In the case of a business using the KBC in a specific application, the level of detail could be much higher. Domain- and inside business knowledge could lead to much more comprehensive KV evaluations.

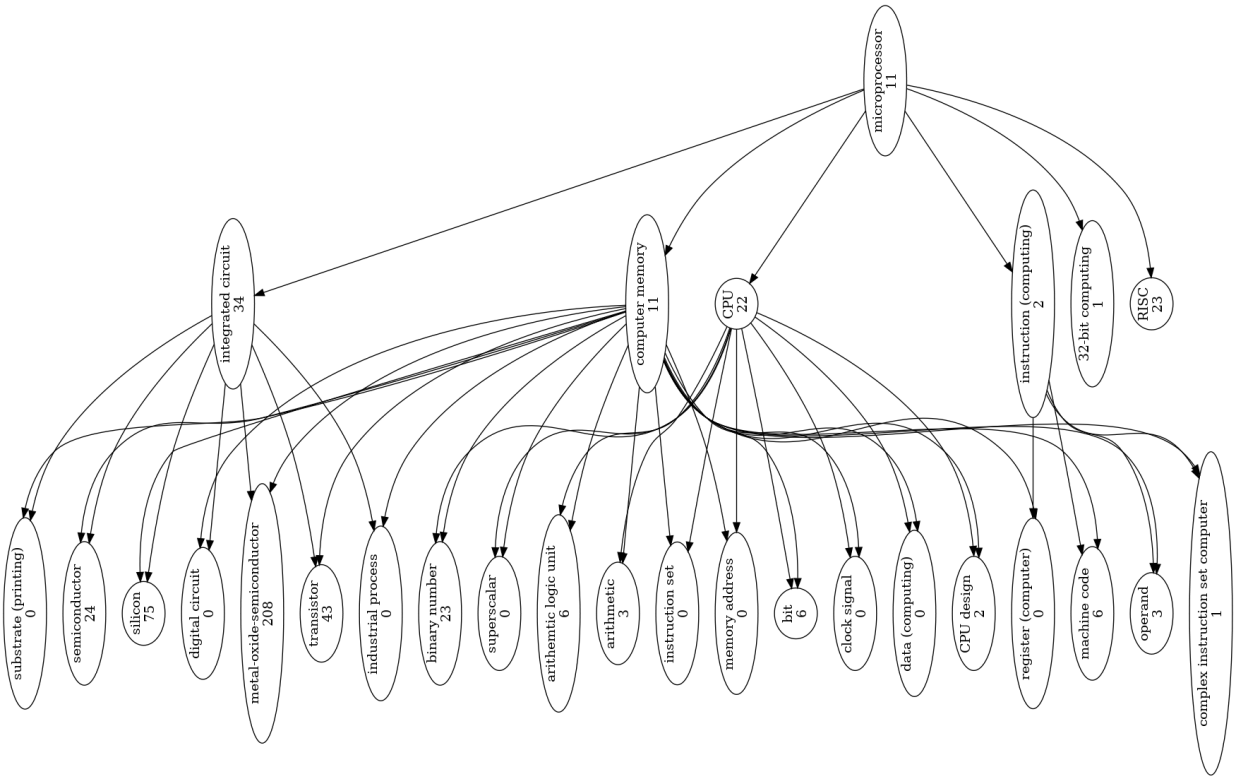

FIGURE 5. *Simplified componentization tree for "microprocessor," generated by Python routine*

REFERENCES

[1] Deshmukh, A.V., Talavage, J. J., Barash, M.M. *Complexity in manufacturing systems, Part 1: Analysis of static complexity* IIE Transactions Vol 30, pp. 645-655 1998.

[2] De Toni, A.F., Nardini, A., Nonino, F. and Zanutto, G. (2001) *Complexity measures in manufacturing systems* Proceedings of the European Conference on Complex Systems November 2005.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* NAACL-HLT (1) 2019: 4171-4186.

[4] *Economic Complexity Index* Wikipedia, Wikimedia Foundation, `https://en.wikipedia.org/wiki/Economic_Complexity_Index` 4 Sep, 2021.

[5] Efatmaneshnik, M., Ryan, M. *A General Framework for Measuring System Complexity* Complexity 21 (S1), pp. 533-546 March 2016.

[6] ElMaraghy, W. H., and Urbancic, R. J. *Assessment of manufacturing operational complexity* Annals of the CIRP, vol. 53, no. 1, pp. 401–406 2004.

[7] ElMaraghy, W. H., and Urbancic, R. J. *Modeling of Manufacturing Process Complexity* Springer, Cham, Switzerland 2006.

[8] Fredendall L. D. and Gabriel T. J. *Manufacturing complexity: a quantitative measure* Proceedings of the POMS Conference, Savannah, GA April 2003.

[9] Frizelle, G. and Woodcock, E. *Measuring complexity as an aid to developing operational strategy* International Journal of Operations and Production Management, Vol. 15, No. 5, pp.26-39 1995.

[10] Hidalgo, C. A. and R Hausmann, R. *The building blocks of economic complexity* Proceedings of the National Academy of Sciences 106.26, pp. 10570–10575 2009.

[11] Hidalgo, C. A. *The dynamics of economic complexity and the product space over 42-year period* CID Working Paper 189, 2009.

[12] Housel, T., Bell, A. A. *Measuring and Managing Knowledge* McGraw-Hill/Irwin, New York City, NY 2001.

[13] Isik, F. *An Entropy-Based Approach for Measuring Complexity in Supply Chains* International Journal of Production Research Volume 48, Issue 12 (online) June 23, 2009.

[14] Lloyd, Seth *Measures of Complexity: A Nonexhaustive List* IEEE Control Systems Magazine 21(4): 7-8 September 2001.

[15] Martínez-Olvera, C. *An Entropy-Based Formulation for Assessing the Complexity Level of a Mass Customization Industry 4.0 Environment* Mathematical Problems in Engineering Volume 2020 Hindawi 2020.

[16] Mealy, P., Farmer, J.D., Teytelboym, A. *Interpreting Economic Complexity* Science Advances, Vol 5, Issue 1 January 9, 2019.

[17] Modrak, V., Marton, D. *Development of Metrics and a Complexity Scale for the Topology of Assembly Supply Chains* Entropy 15(10), pp. 4285-4299 2013.

[18] Soares LB, FitzGerald N, Ling J, Kwiatkowski T. *Matching the blanks: Distributional similarity for relation learning* Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, pp. 2895-2905 July 28 - August 2, 2019.

[19] Sönmez, Ö.E. and Koç, V. T. *On quantifying manufacturing flexibility: an entropy based approach* Proceedings of the World Congress on Engineering, London, UK July 2015.